**Git Bash**

Git Bash is an application for Microsoft Windows environments which provides an emulation layer for a Git command line experience.

* Bash is an acronym for **Bourne Again Shell**.
* A shell is a terminal application used to interface with an operating system through written commands.

**Git Bash Download for windows**

<https://git-scm.com/download/win>

**Linux Commands**

Pwd: Path of the current working directory

Mkdir: Make a directory

Cd: Change directory

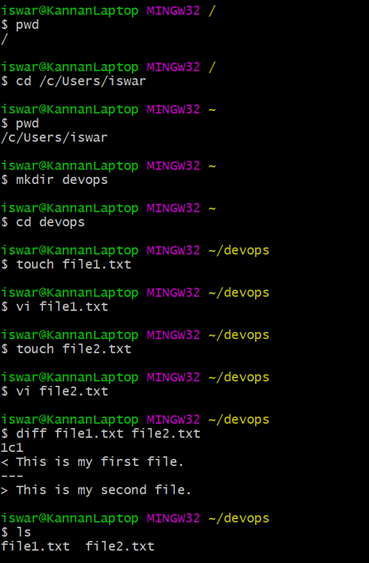
Touch: Used to create blank file

Vi: It is the most popular text editor in Linux.

1. To enter vi, type: vi filename
2. To enter insert mode, press : shift i
3. Type in the text: This is easy
4. To leave insert mode and return to command mode, press: <Esc>
5. In command mode, save changes and exit vi by type shift: wq shift1 <Return> You are back at the Unix prompt.

Diff: diff command compares the contents of two files line by line

Ls: list the directory



Echo Hi > file1.txt : Replace the entire file with the message “Hi”

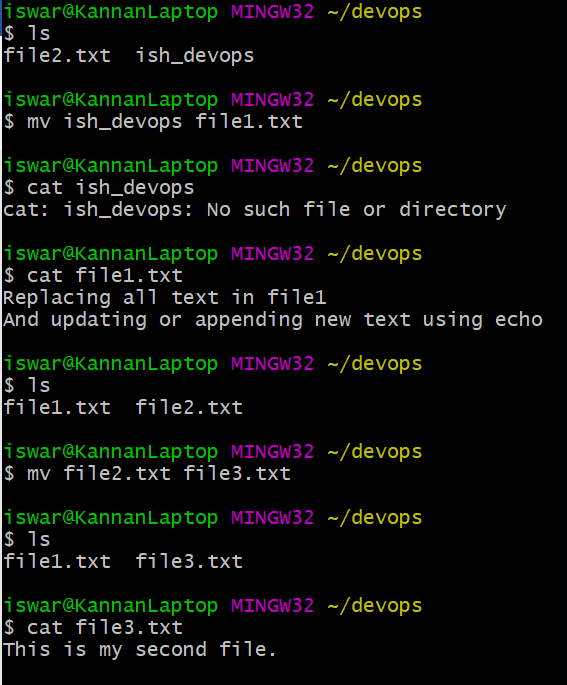
Echo This is linux command>>file1.txt : Appends the text with the previous file. “Hi This is linux command”

Cat: Shows the content of the file



Cp: Copy files from the files from current directory to the specified directory

Mv: Move the file from current directory to the specified directory



Rm: Used to delete file

Rmdir: Used to remove directory

Grep word file.txt: Used to find the word in the file.txt



**Github**

* GitHub offers a cloud-based Git repository hosting service.
* Essentially, it makes it a lot easier for individuals and teams to use Git for version control and collaboration.

1.git config

To set the basic configurations on github like your name and email.

2.git config --global user.name "IswaryaG1017"

Sets configuration values for your user name on git.

3.git config --global user.email "iswaryak1017@gmail.com"

Sets configuration values for your user email on git.

4.git init

To create a local git repository for us in our store folder. This will help to manage the git commands for that particular repository.

5.git add Readme.txt

To add a file Readme.txt to the staging area to track its changes.

6.git commit -m “Created a Readme.txt”

To commit our changes (taking a snapshot) and providing a message to remember for future reference.

7. git log

To check the history of commits for our reference.

9.git status

To see what’s changed since last commit. It shows all the files that have been added and modified and ready to be committed and files which are untracked

10.git diff

To figure out what changes you made since last commit.

11.git checkout –license

To Blow away all changes since the last commit of the file.

12.git remote add origin https://github.com/madaan123/MyAlgorithms.git

These commands make a bookmark which signifies that this particular remote refers to this URL.

This remote will be used to pull any content from the directory and push our local content to the global server.

13.git remove rm

To remove a remote from our local repository.

14.git push -u origin master

To push all the contents of our local repository that belong to master branch to the server (Global repository).

15.git clone https://github.com/madaan123/MyAlgorithms.git

To clone or make a local copy of the global repository in your system

(git clone command downloads the repository and creates a remote named as origin which can be checked by command – git remote -v).

16.git branch Testing

To create a new branch named as Testing.

17.git branch

To see all the branches, present and current branch that we are working on.

18.git checkout Testing

To switch to branch Testing from master branch.

19.ls

To see directories and files in the current directory.

20.git merge Testing

To merge Testing branch with master branch.

21.git branch -d Testing

To delete Testing branch.

22.git checkout -b admin

To create a new branch admin and set it as current branch.

23.git branch -r

To look at all the remote branches.

24.git branch -D Testing

To forcefully delete a branch without making commits.

25. git pull

Fetch and merge any commits from the tracking remote branch.

**Different ways to use add command:**

1.git add

To add a specific list of files to staging area.

2.git add --all

To add all files of current directory to staging area.

3.git add \*.txt

To add all text files of the current directory to staging area.

4.git add docs/\*.txt

To add all text files of a particular directory(docs) to staging area.

5.git add docs/

To add all files in a particular directory(docs) to staging area.

6.git add “\*.txt”

To add text files of entire project to staging area.

**Linux user management (File Permission)**

Sudo: Sudo is the command enables you to perform tasks that require administrative or root permissions.

Chmod: Used to change the read, write and execute permissions of the files or directories

1. **We need to be a root user in order to add a user**
2. **EC2 User by default is a root user**
3. **Adding new user**

[ec2-user@ip-172-31-21-149 ~]$ sudo adduser user2

[ec2-user@ip-172-31-21-149 ~]$ sudo passwd user2

passwd: all authentication tokens updated successfully.

1. **Listing users**

[ec2-user@ip-172-31-21-149 ~]$ ls /home/

ec2-user user1 user2

1. **Switching users to user1**

[ec2-user@ip-172-31-21-149 ~]$ su - user1

Password:

[user1@ip-172-31-21-149 ~]$logout

1. **To be as a root user**

[ec2-user@ip-172-31-21-149 ~]$ sudo -i

[root@ip-172-31-21-149 ~]# visudo

1. **In the visudo file search for wheel access(for password escalation access)**

## Same thing without a password

# %wheel ALL=(ALL) NOPASSWD: ALL --🡪 uncomment this

[root@ip-172-31-21-149 ~]# usermod -aG wheel user2

[root@ip-172-31-21-149 ~]# logout

[ec2-user@ip-172-31-21-149 ~]$ sudo usermod -aG wheel user2

[ec2-user@ip-172-31-21-149 ~]$ su - user2

Password:

Last failed login: Tue Feb 22 03:12:05 UTC 2022 on pts/0

There were 2 failed login attempts since the last successful login.

[user2@ip-172-31-21-149 ~]$ sudo yum update

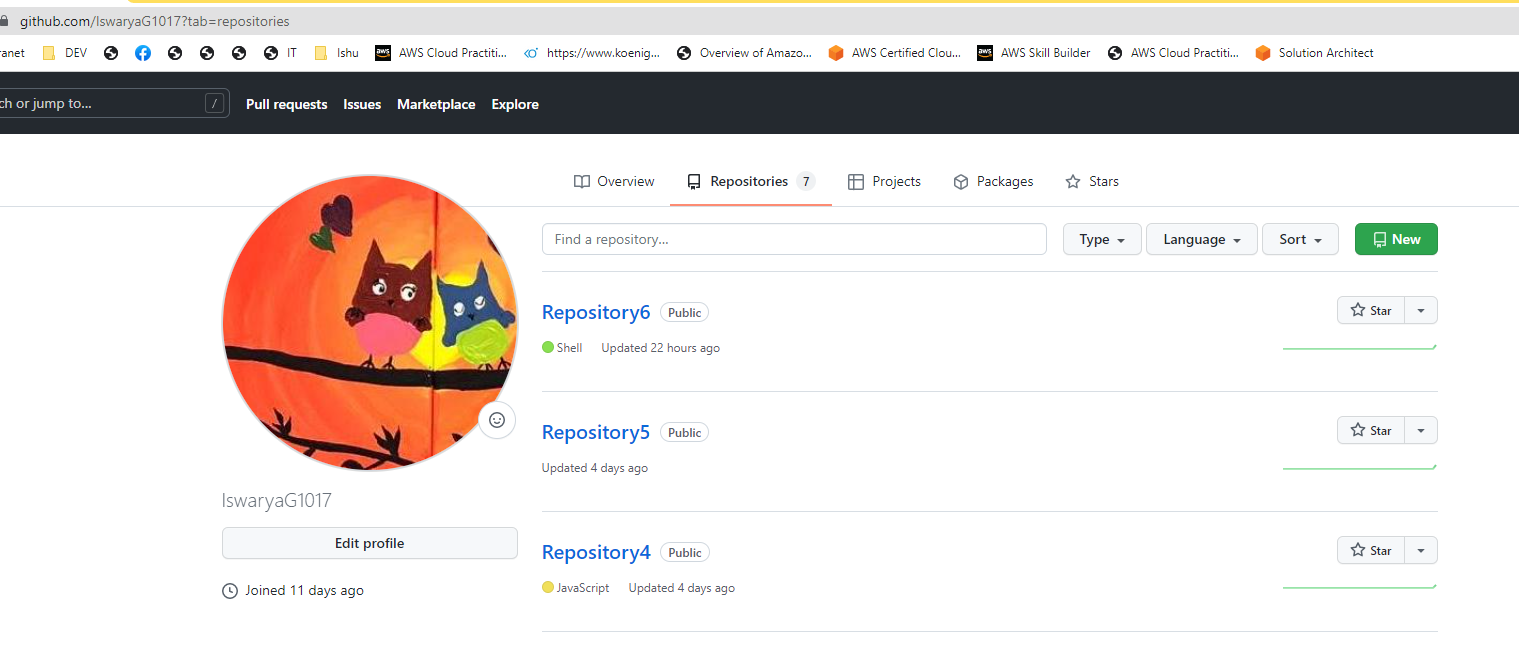
[user2@ip-172-31-21-149 ~]$ sudo adduser user2\_test

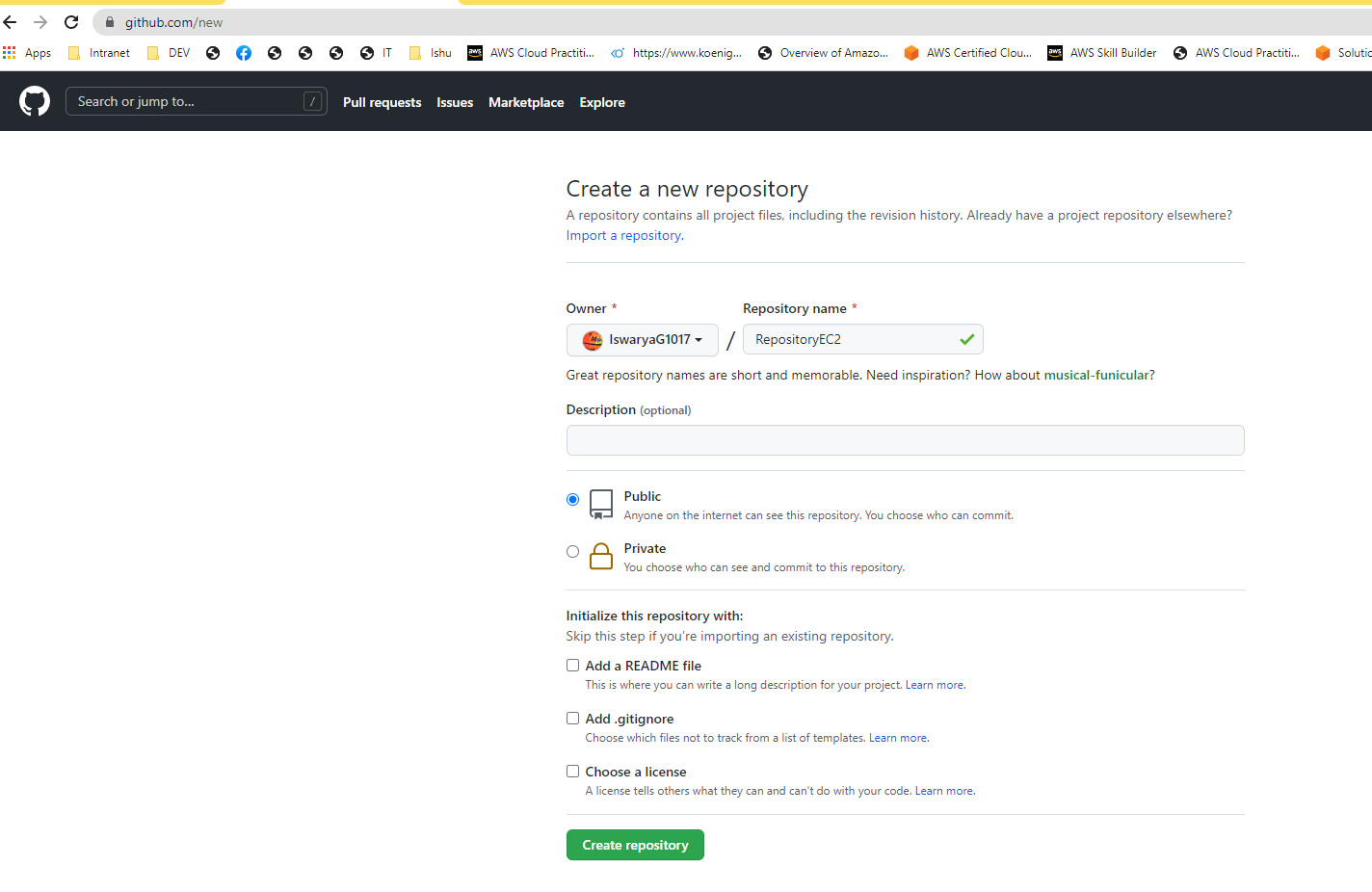
[user2@ip-172-31-21-149 ~]$ sudo passwd user2\_test

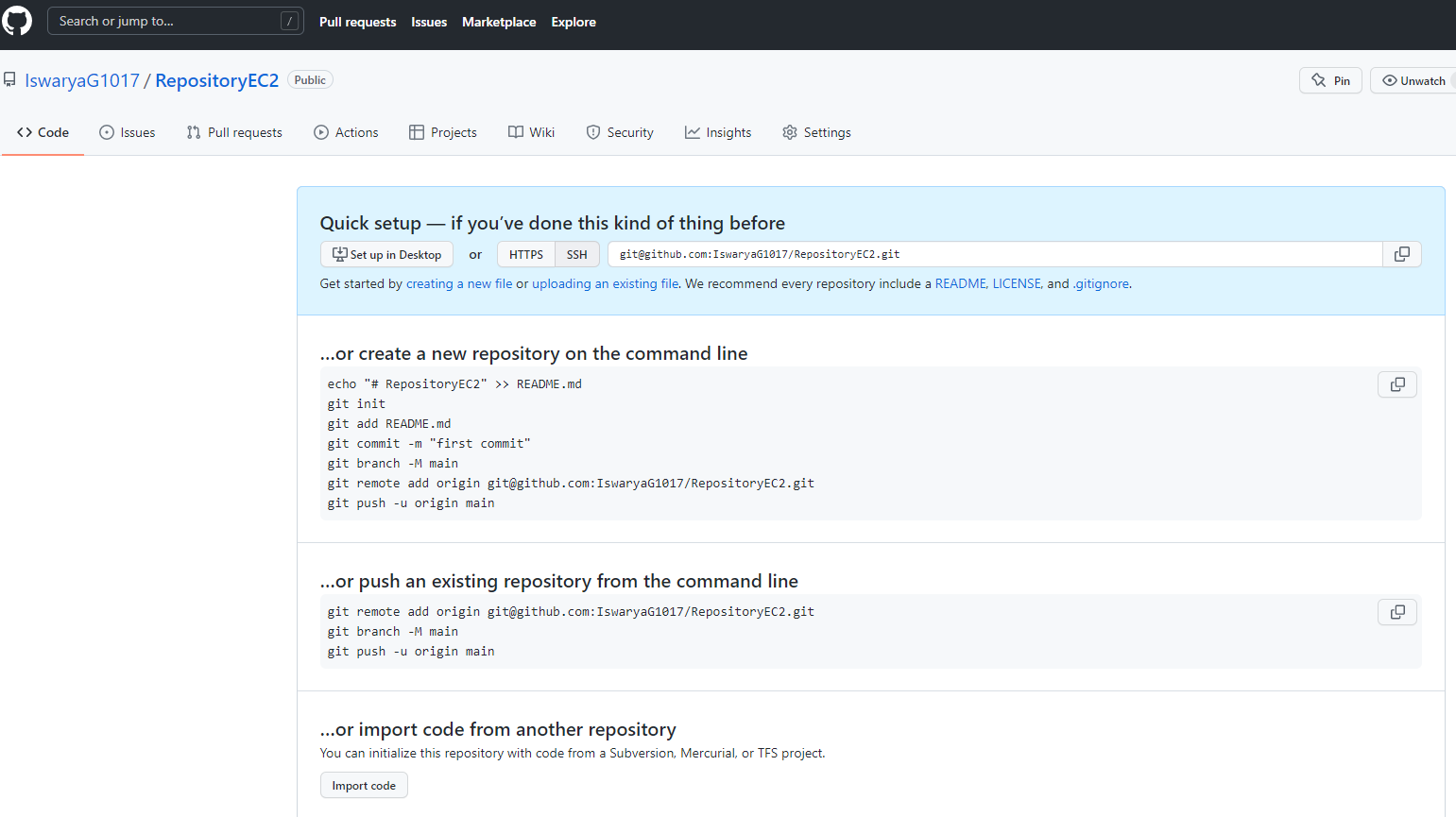
[user2@ip-172-31-21-149 ~]$ su - user2\_test

**Login to github:**

[**https://github.com/login**](https://github.com/login)

****

****

****

**EC2 Instances**

1. An Amazon EC2 instance is a virtual server in Amazon's Elastic Compute Cloud (EC2) for running applications on the Amazon Web Services (AWS) infrastructure.
2. Users can select an AMI provided by AWS, the user community or through the AWS Marketplace. Users also can create their own AMIs and share them.

|  |  |
| --- | --- |
| **Instance Family** | **Some Use cases** |
| General Purpose (t2, m4, m3) | * Low traffic websites and web applications * Small databases and mid-size databases |
| Compute Optimized (c4, c3) | * High performance front-end fleets * Video encoding |
| Memory Optimized (r3) | * High performance databases * Distributed memory caches |
| Storage Optimized (i2, d2) | * Data Warehousing * Log or data-processing applications |
| GPU instances (g2) | * 3D application streaming * Machine Learning |

**EC2 Instance Pricing**

**On-Demand**

* With on demand EC2 instances you only pay for compute capacity you use by hours or by seconds depending on the instance types.

**Amazon EC2 Savings Plans**

* AWS offers Savings Plans for several compute services, including Amazon EC2.
* Amazon EC2 Savings Plans enable you to reduce your compute costs by committing to a consistent amount of compute usage for a 1-year or 3-year term.
* This term commitment results in savings of up to 66% over On-Demand costs.

**Reserved Instances**

* Reserved instances involve making a compute capacity reservation with AWS on contractual basis. The contract for the reserved compute capacity could be anywhere between 1-3 years.
* Upfront cost is involved when selecting this type of EC2 instance.

### **There are three types of Reserved Instances:**

* **Standard reserved instance:** It provides a discount of up to 75% off on demand. The more you pay upfront and larger the duration of contract the larger the discount.
* **Convertible reserved instance:**It provides a discount of up to 54% off on demand. It provides the feature that has the capability to change the attributes of RI as long as the exchange results in the creation of Reserved Instances of equal or greater value.
* **Scheduled reserved instance:** Scheduled Reserved Instances are available to launch within the specified time window you reserve. It allows you to match your capacity reservation to a predictable recurring schedule.

**Spot Instances**

* Spot EC2 Instance lets bid for the spare compute capacity from Amazon at heavy discounts of up to 90% of the On-Demand price.

**Dedicated Hosts**

* Dedicated Instances are physical EC2 servers that are dedicated for your use.

1. Create a User (EC2 instance)

2. Add users (EC2)

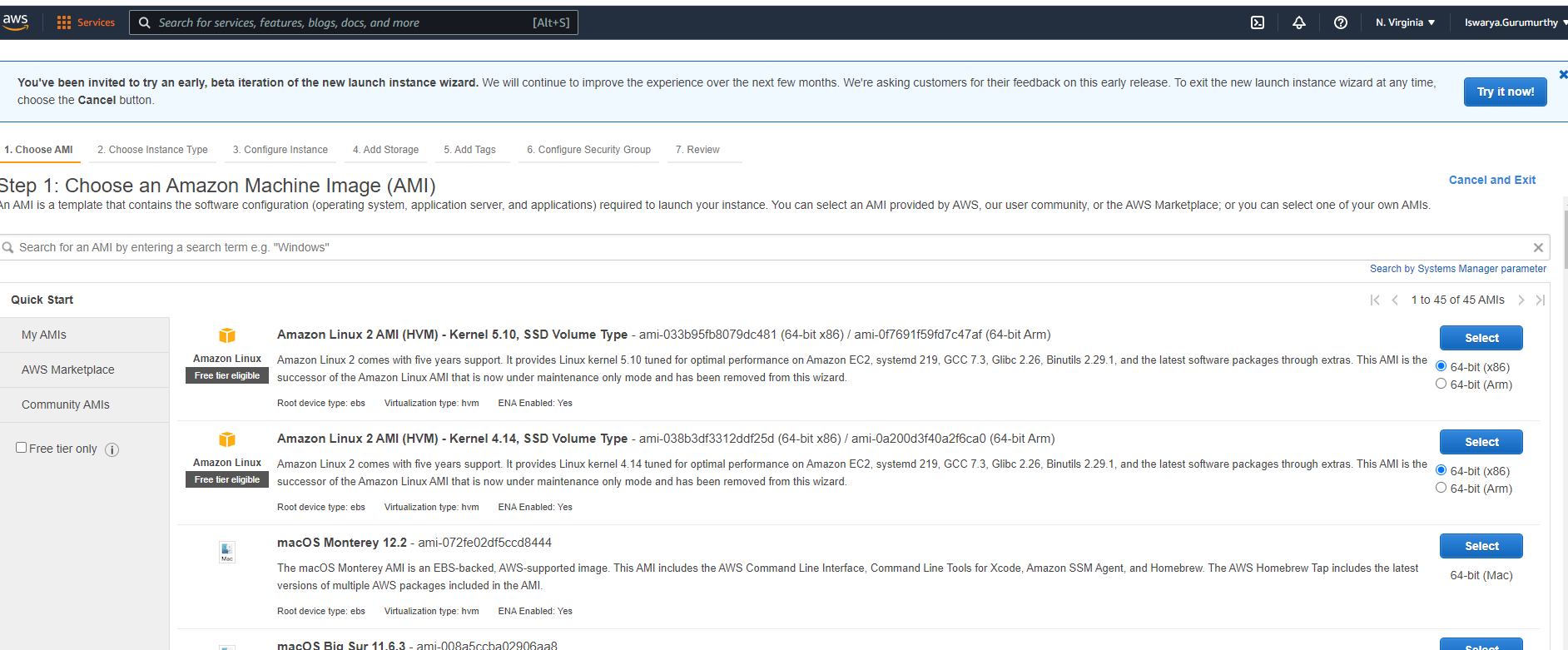
3. Login as new user – EC2 user

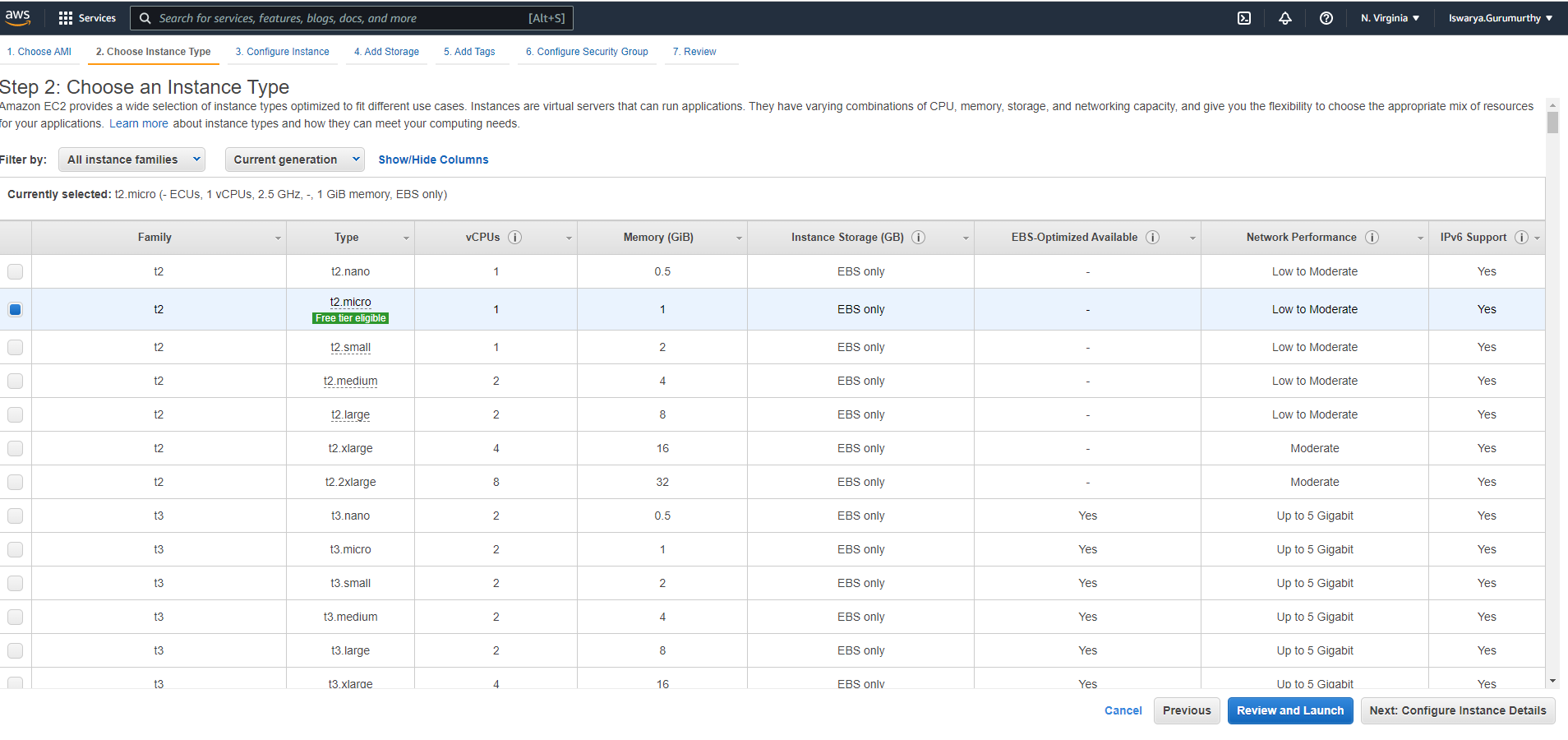
4. Generate a ssh key, add public github

5. Clone repo1 for user1 and add 3 files

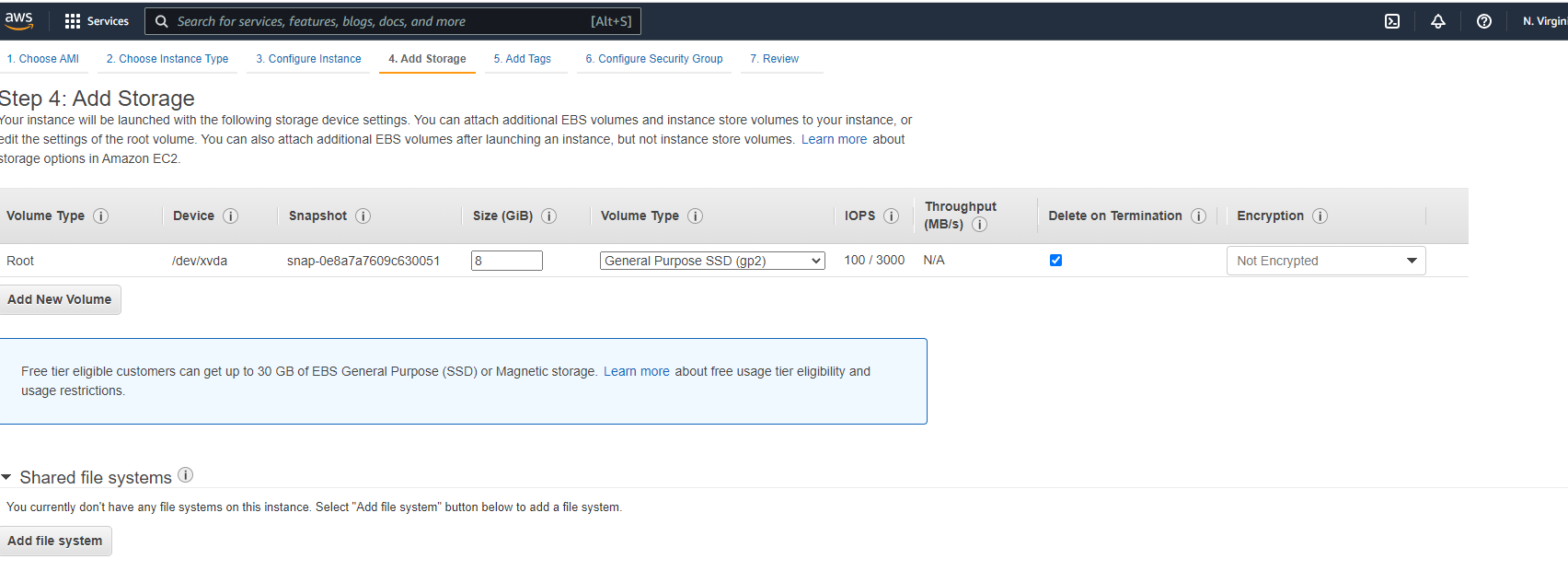
**Launch instance**



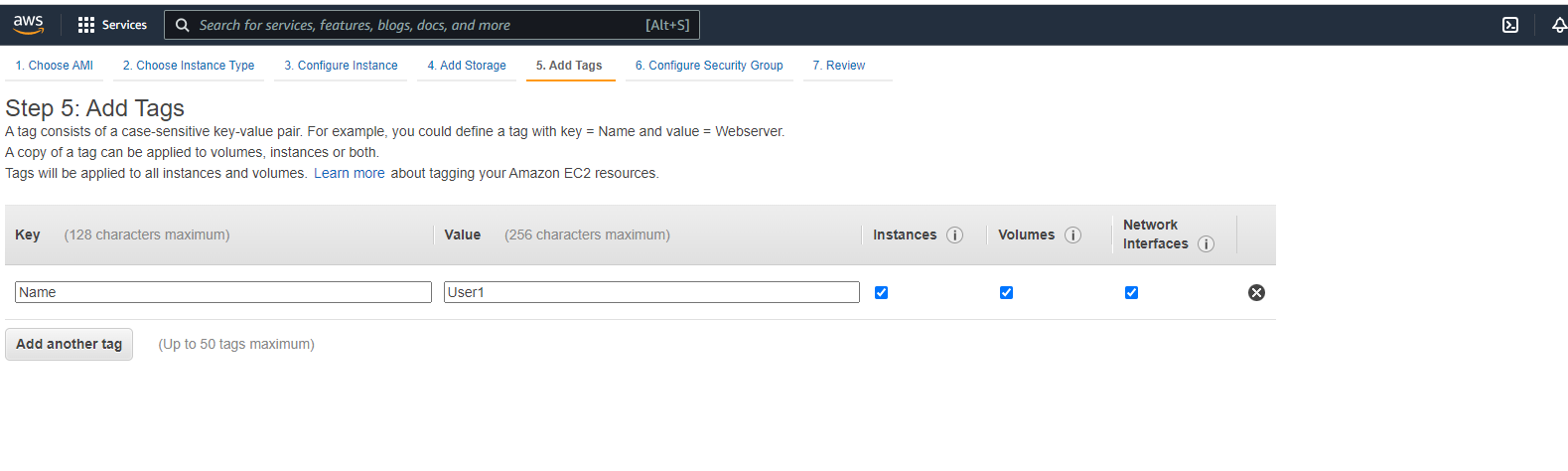
**Choose AMI (Amazon Machine Image)**

**Choose and Instance Type**

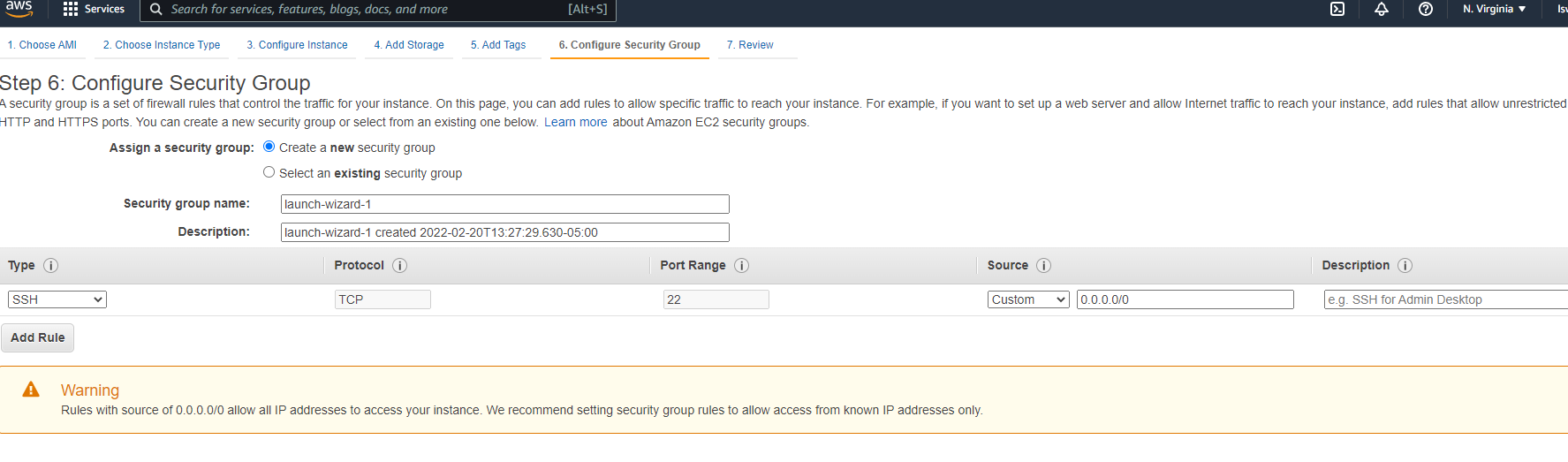
**Add Storage**



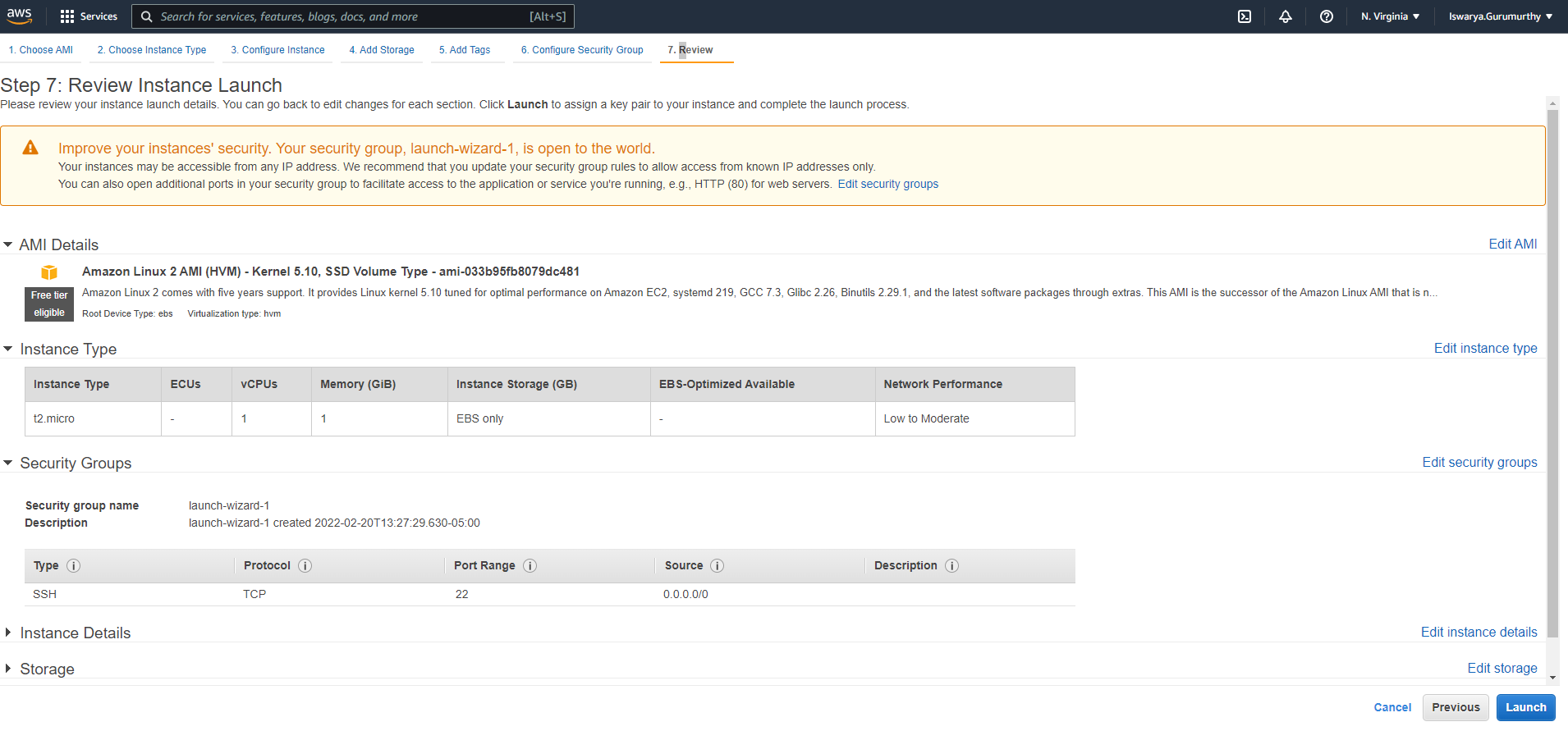
**Add Tags**

****

**Add Security Group**

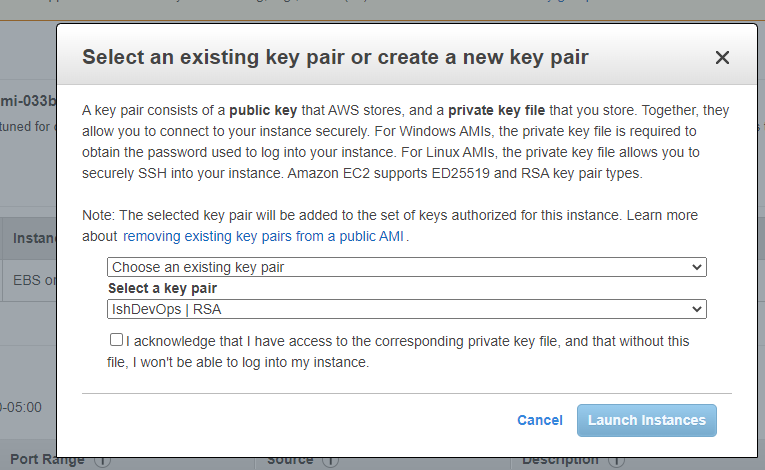
****

**Review & click Launch**

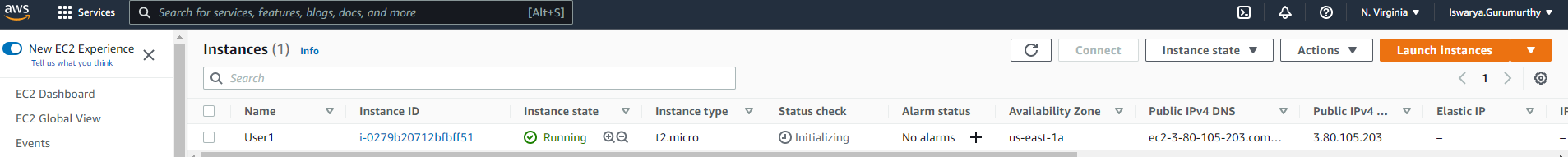
****

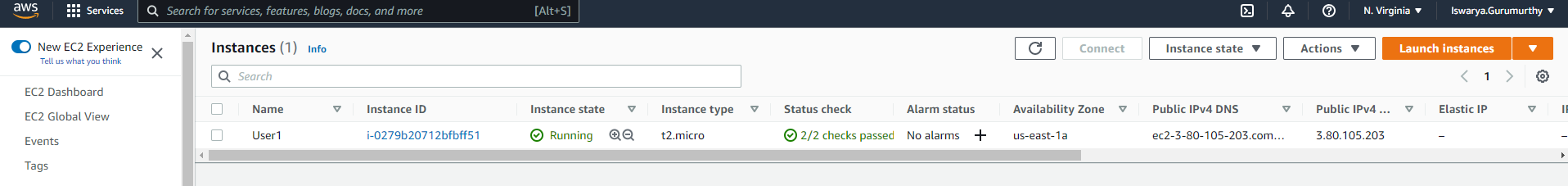
**Select an existing key pair**

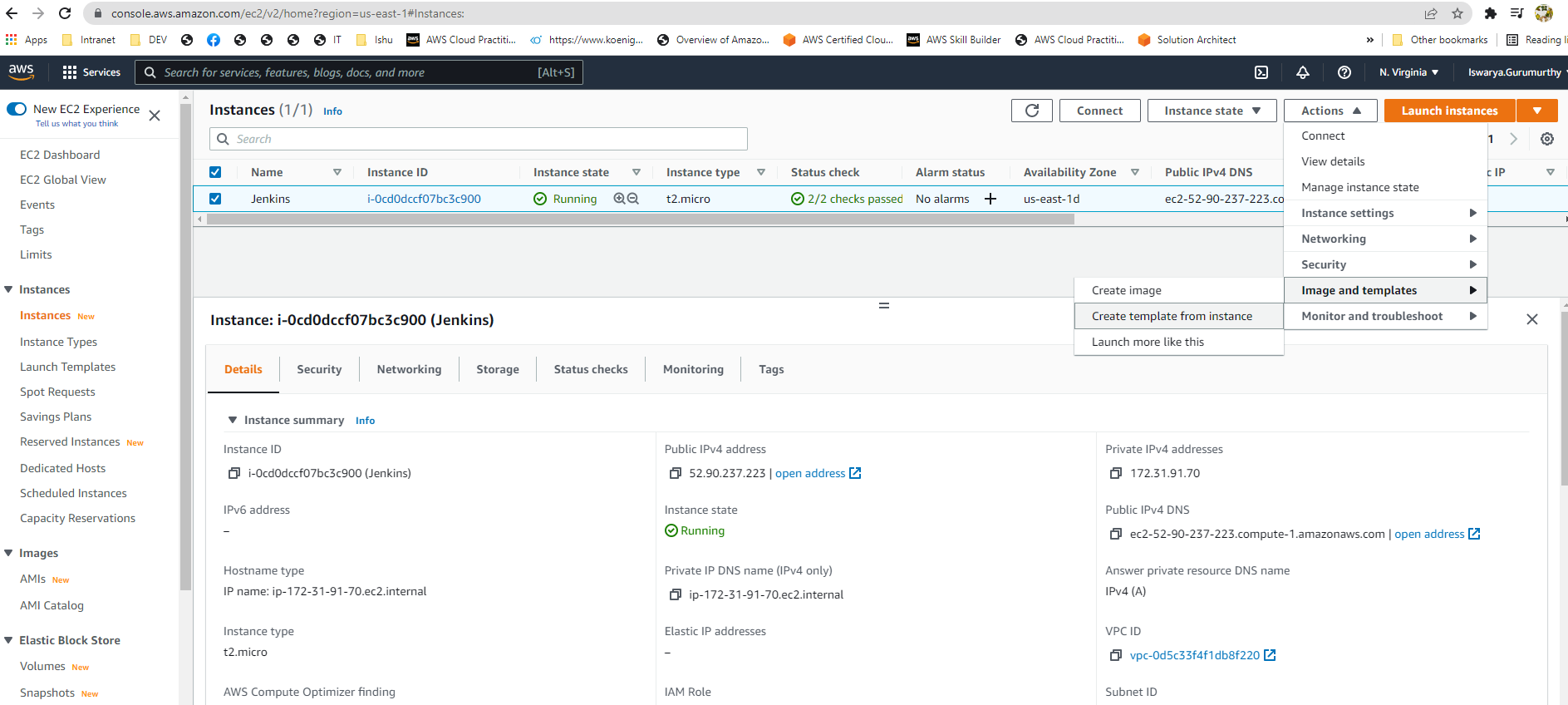
**check the acknowledge check box and click launch**

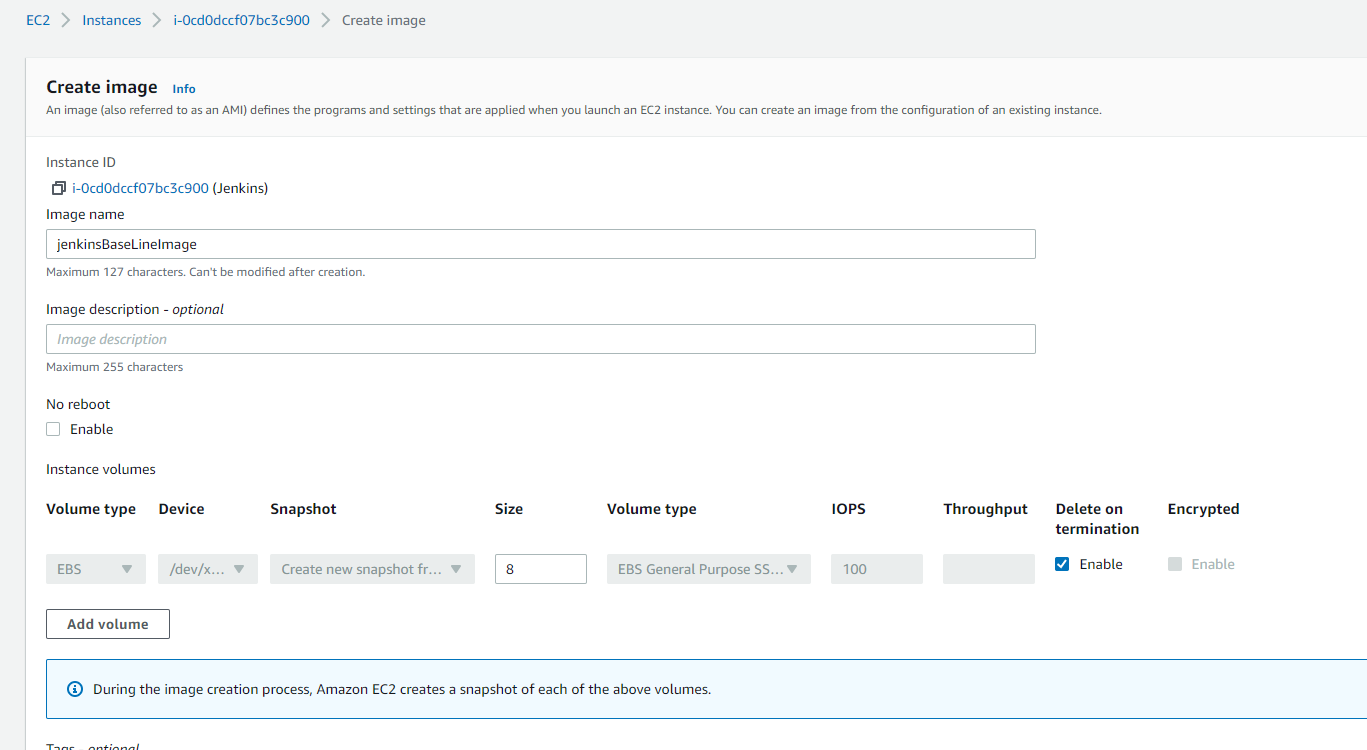












Create EC2 instance using above

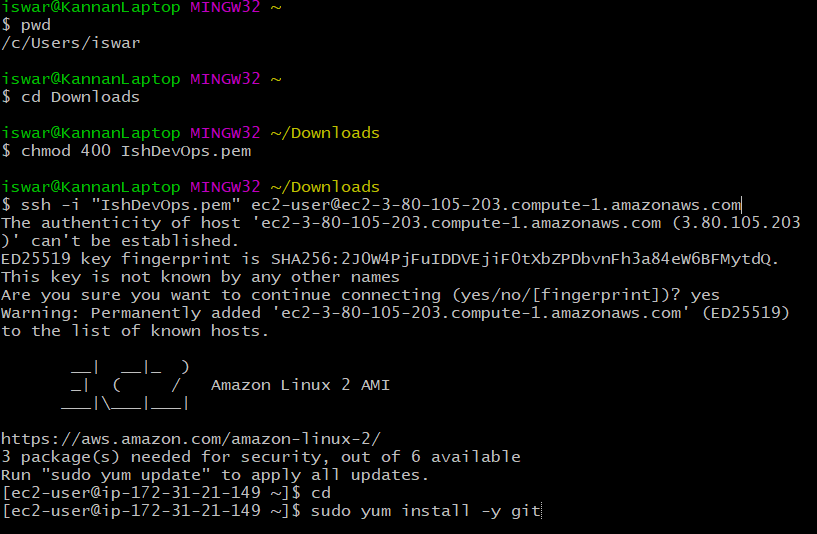
steps:

1. Login to AWS management Console
2. Launch an EC2 instance
3. Create an Ec2 instance
4. Goto Ec2
5. Launch Instance
6. Amazon linux2 AMI(HVM)
7. t2Micro-Free tier Eligible
8. No. of instances 1
9. Root - 8Gb
10. Add tags -> Key: name Value: Webserver
11. Security Group
12. click Review and launch
13. Launch
14. Choose an existing key pair (IshDevOps)
15. I acknowledge
16. Click launch

-------------------------------------------------------------------------------------------

1. Open Terminal/Gitbash
2. cd /c/Users/iswar
3. pwd
4. ls
5. cd Downloads
6. Again go back to EC2 click on the instance and connect
7. Goto the SSH client tab
8. copy chmod 400 IshDevOps.pem
9. copy ssh -i "IshDevOps.pem" ec2-user@ec2-3-80-200-71.compute-1.amazonaws.com
10. paste in terminal/gitbash
11. chmod 400 IshDevOps.pem
12. ssh -i "IshDevOps.pem" ec2-user@ec2-3-80-200-71.compute-1.amazonaws.com
13. Are sure you want to continue connecting? Yes
14. Cd
15. Sudo yum install git

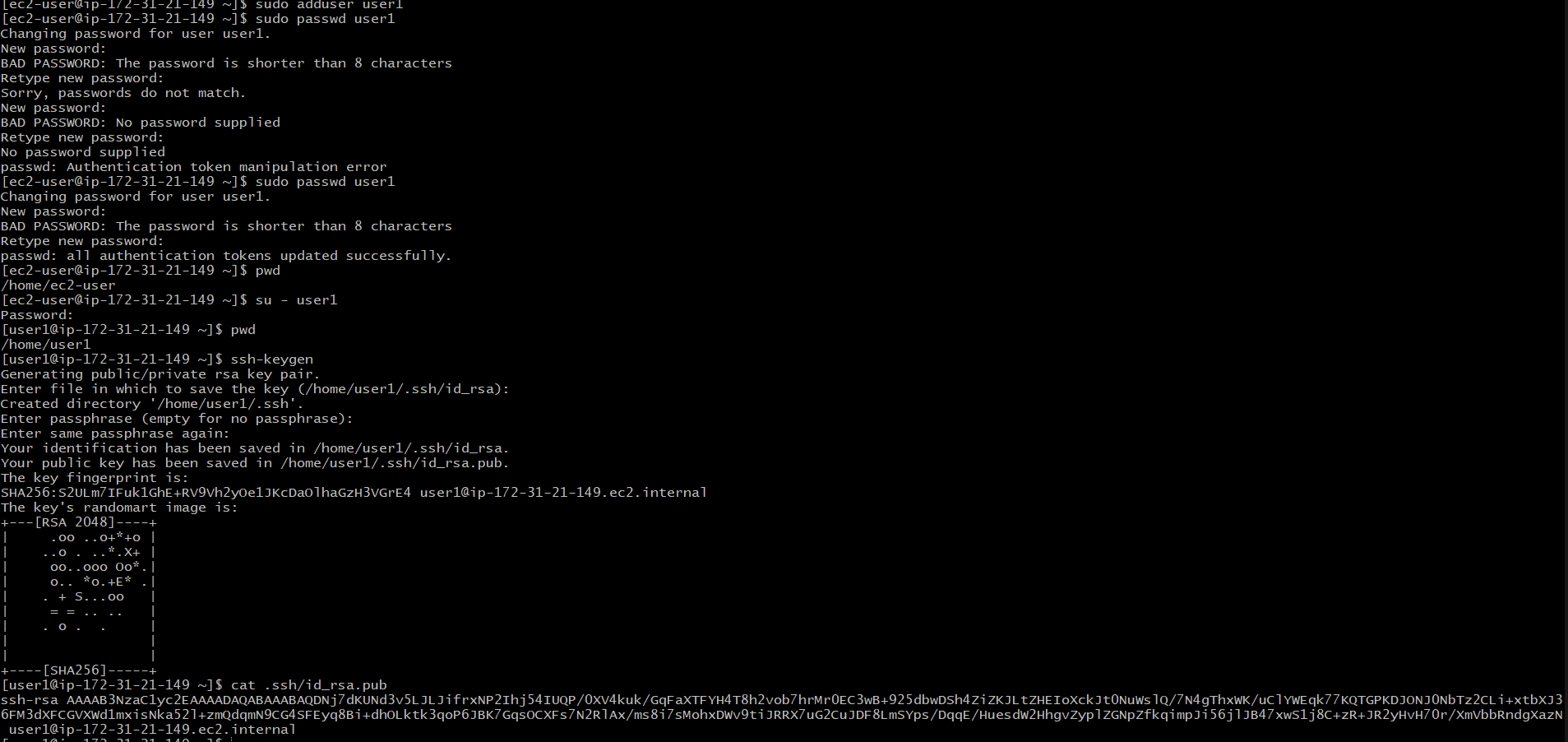
----------------------------------------------------------------------------------------------



1. sudo adduser user1
2. sudo passwd user1
3. 123456
4. press Enter,Enter and Enter
5. su – user1
6. pwd

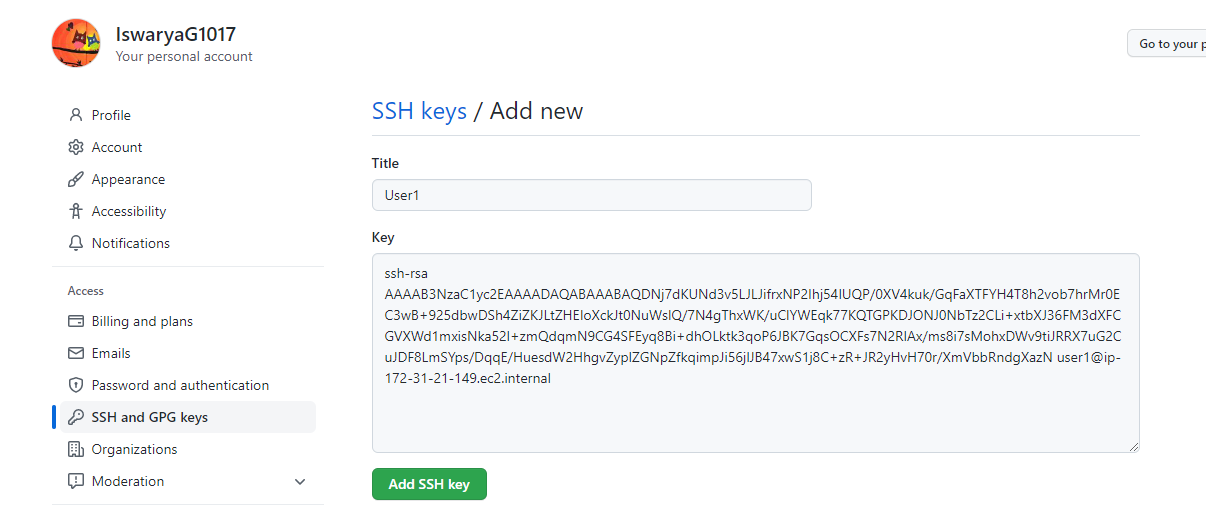
/home/user1

1. ssh-keygen (Enter,Enter and Enter)
2. cd .ssh/ or cat .ssh/id\_rsa.pub
3. ls
4. cat id\_rsa.pub

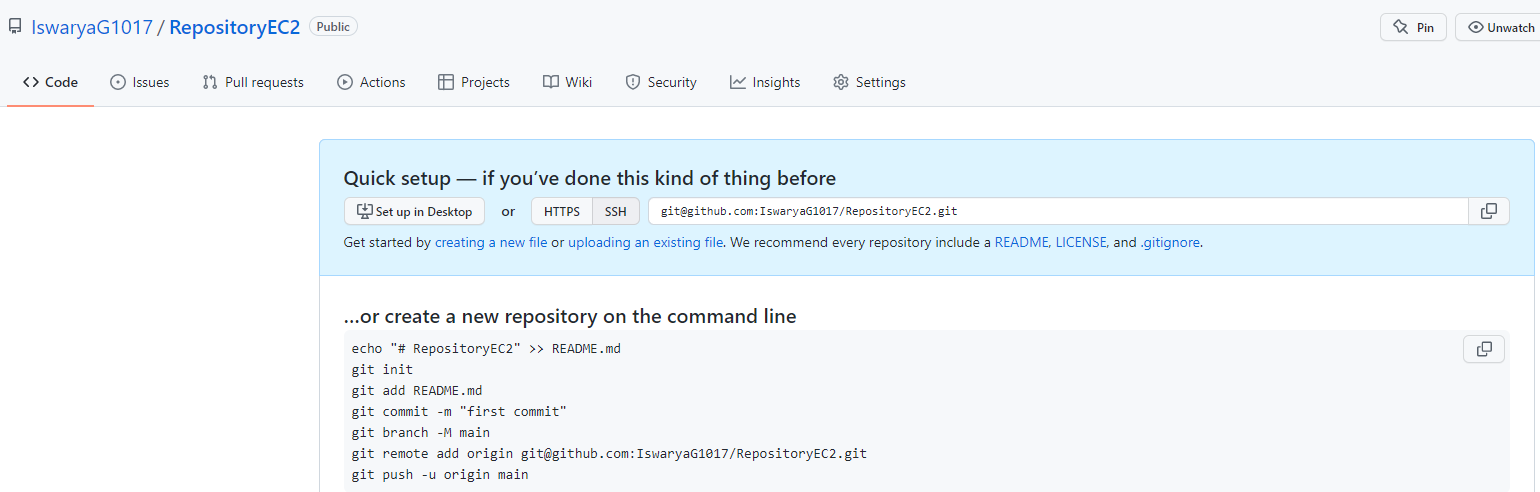


Goto github-> settings-> SSH and GPG keys -> NewSSH keys -> Paste here

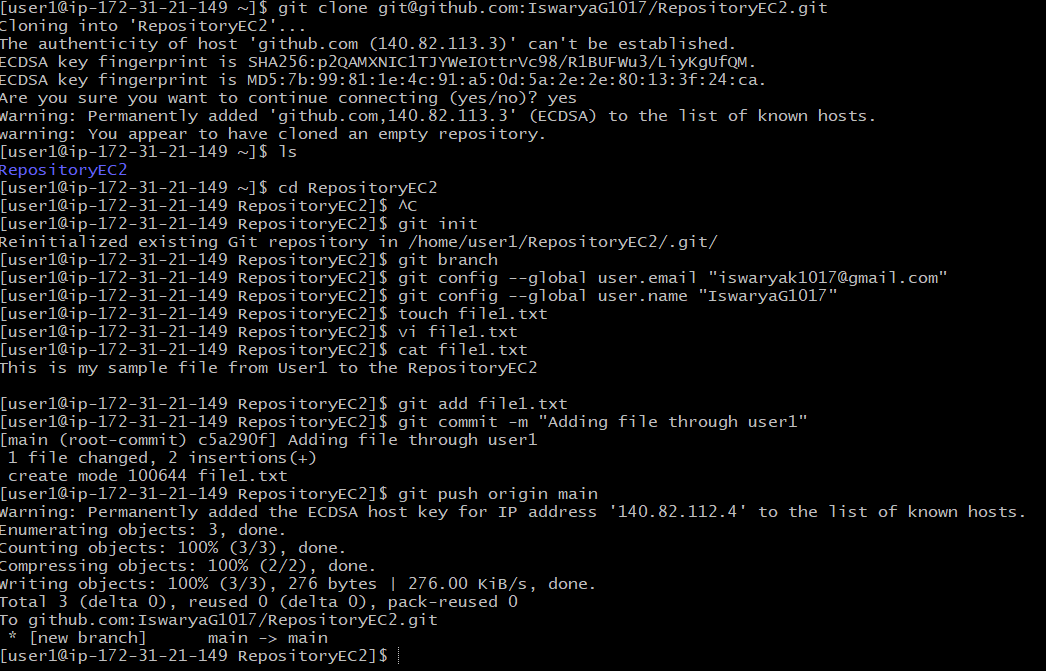
1. ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC8mQ5eabNhzouRdVvr/F8WM1eV25pNcNsMV6ktMFwB7Hs/5oaO7teT+CGjcpSpiFr0zNThNA7R9LELNtmFBKF8gJ0eRqQ9J3Lf/cc/boXUS9Bt+1vEANh40mcF1bYXKxDer2y4LgscBRLyLNSJq2PpiOBeJkuA9qFjb2QIlvARRVCwKxl+Mcmvts3EHEXzuoWmbCKsHDMoEj1clrN9tsoaaJXbu51Sz4RmROoHmXNhOkzxmSz01rDFdnG7CXoyot7Rnlvzph6se5+Wrew81dBe+so+JsqWliU8SGCamaECT9bWtluQBMGAf0O75svQwkwfByJVuOr6/mXL3cDI0lD5 usertwo@ip-172-31-86-187.ec2.internal
2. and press Add SSH key
3. Confirm ur github password

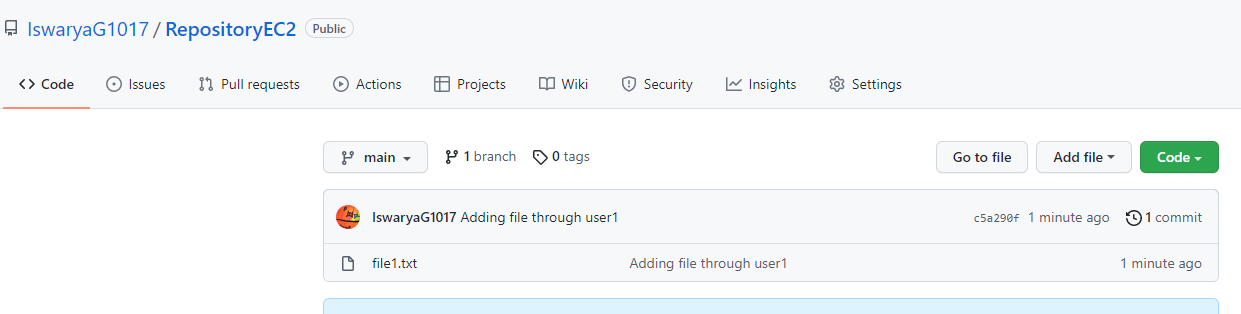


1. go to new repository - copy ssh



1. Goback to terminal
2. cd
3. git clone [git@github.com:IswaryaG1017/RepositoryEC2.git](mailto:git@github.com:IswaryaG1017/RepositoryEC2.git)
4. Are you sure?
5. yes
6. ls
7. cd RepositoryEC2
8. git init
9. git branch
10. git config --global user.email "iswaryak1017@gmail.com"
11. git config --global user.name "IswaryaG1017"
12. touch file1.txt
13. vi file1.txt
14. shift+i
15. Type something
16. esc shift: **wq** shift+1
17. git add file1.txt
18. git commit -m "Adding file through devops user1"
19. git push origin main
20. git push -u origin main





--------------------------------------------------------------------------------------------------------------------

Create new branch from base branch

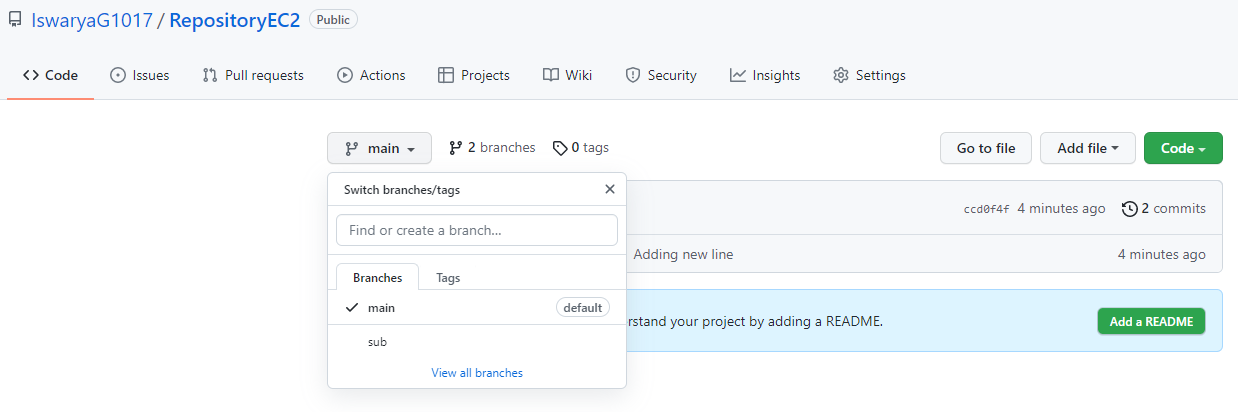
Git checkout -b new branchname

Add same file -file1.txt with changes

Create a pull request from your new branch to your base branch

Take a screenshot of pull req and merge

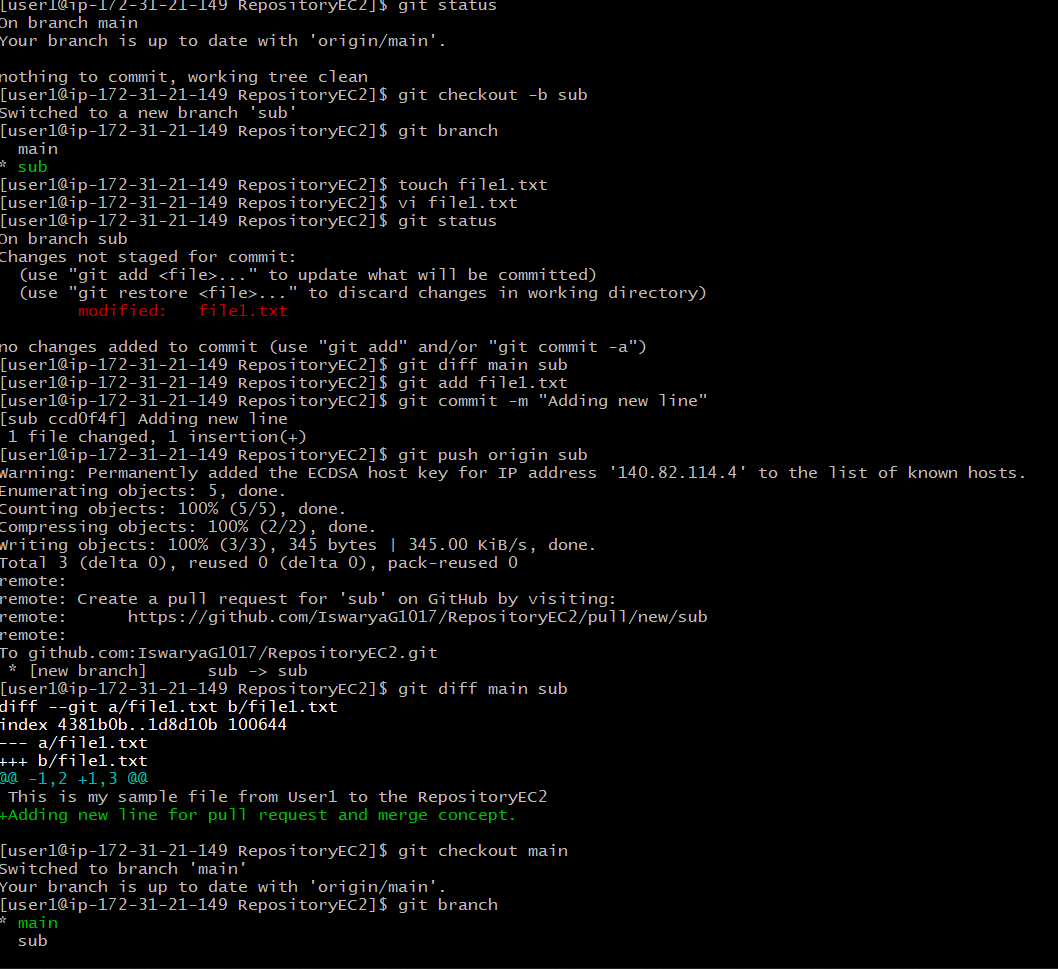
1. git status
2. git branch sub or git checkout -b sub

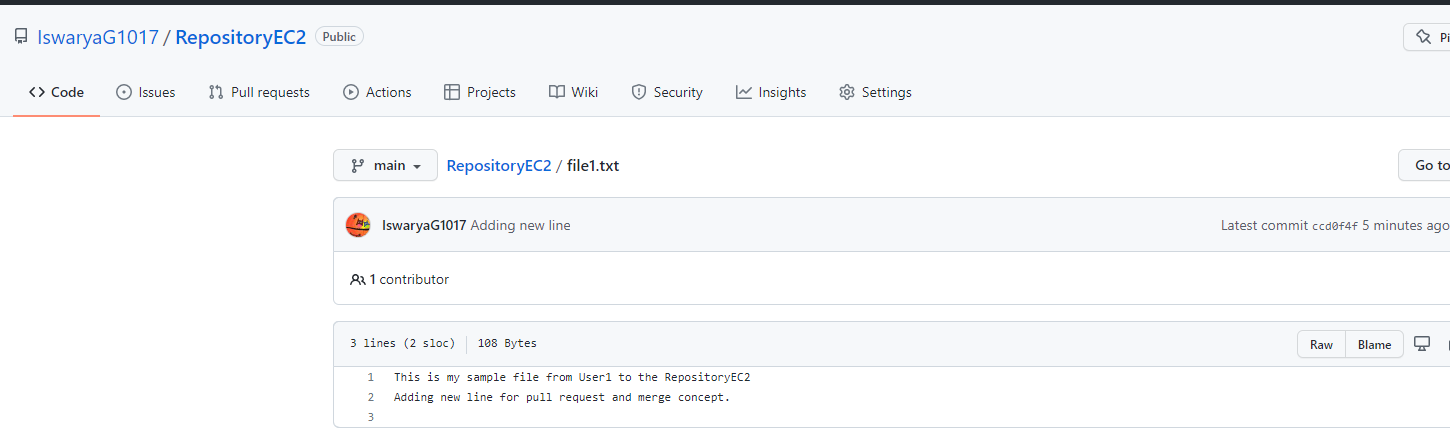


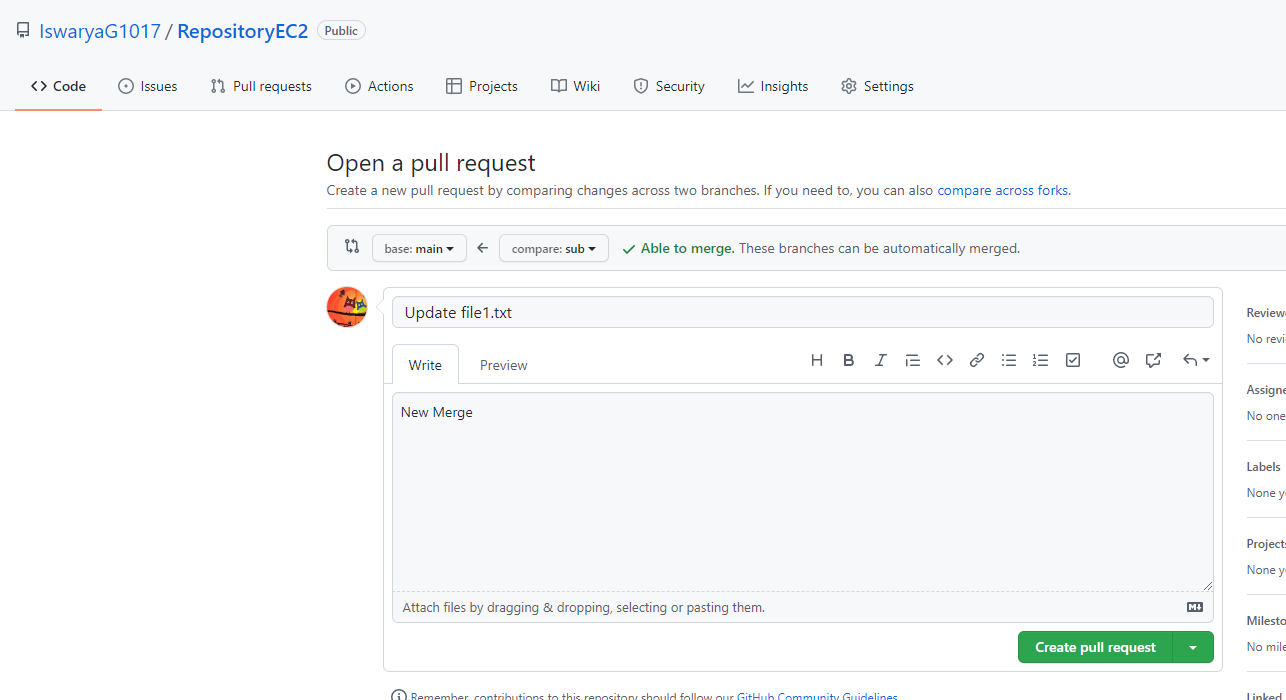
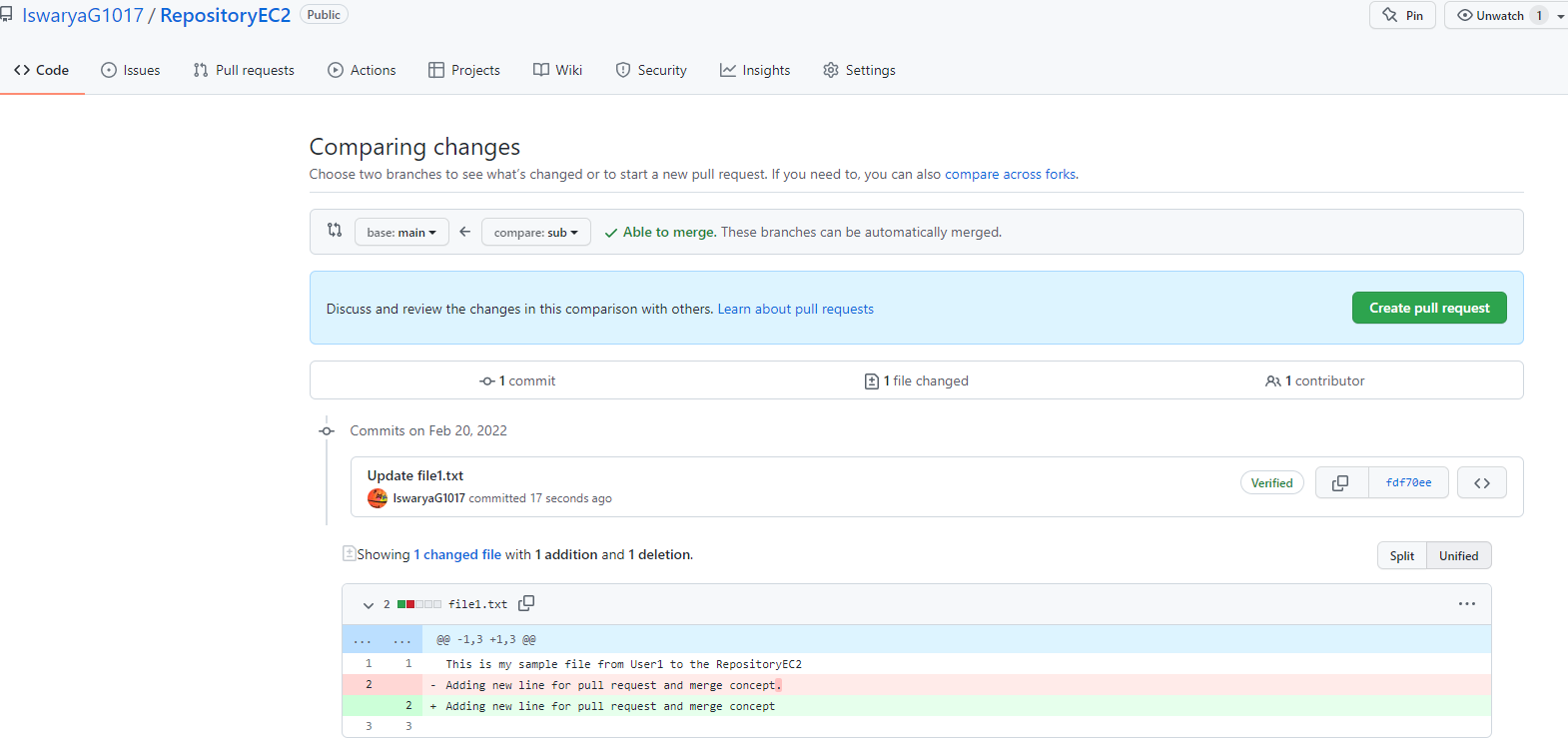
1. git checkout sub
2. git branch or git branch --list
3. touch file1.txt
4. vi file1.txt
5. git status
6. git add file1.txt
7. git commit -m "Adding new function"
8. git push origin sub
9. git diff main sub
10. git checkout main
11. git branch

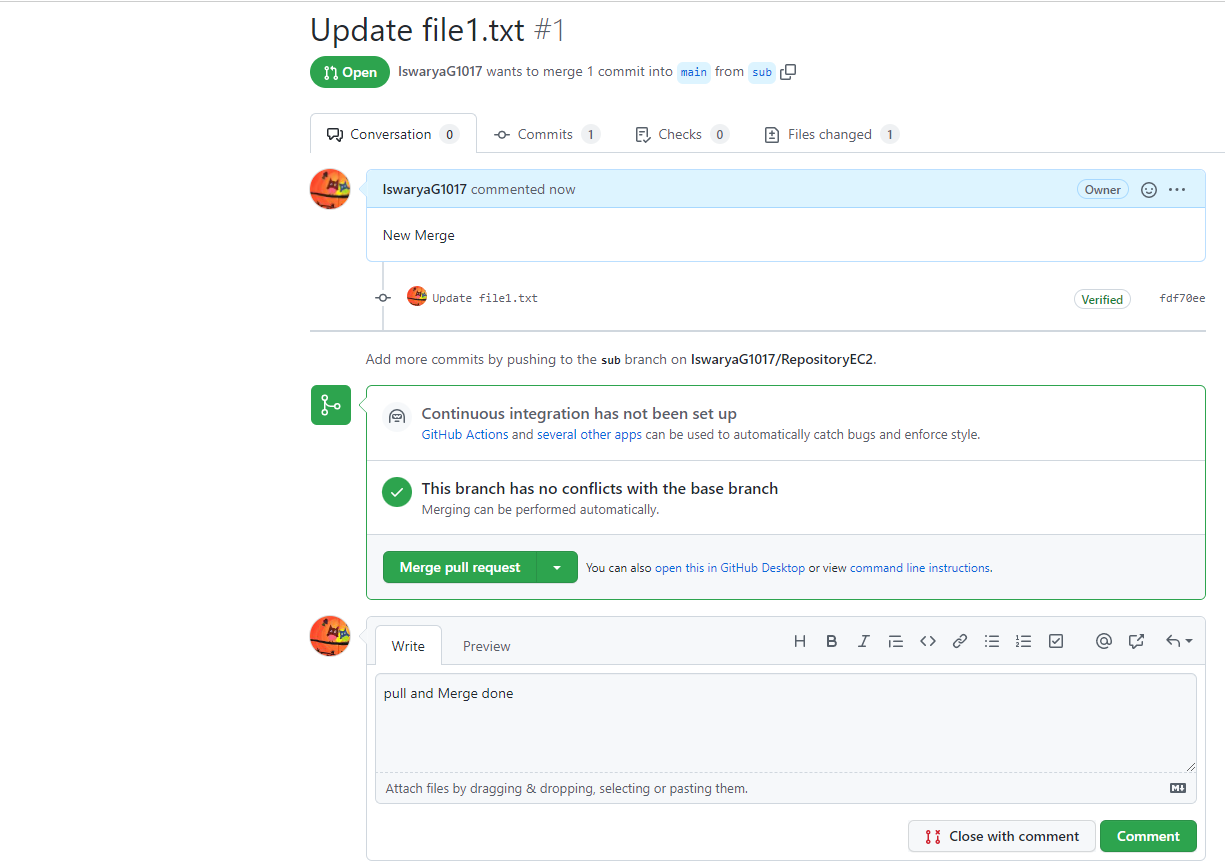
* sub
* \*main

1. git merge sub
2. git status
3. git push origin main
4. git pull
5. Exit









**Jenkins**

## What is IAM?

AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS resources. It enables you to create and control services for user authentication or limit access to a certain set of people who use your AWS resources.

### Users

An IAM user is an identity with an associated credential and permissions attached to it. This could be an actual person who is a user, or it could be an application that is a user. With IAM, you can securely manage access to AWS services by creating an IAM user name for each employee in your organization. Each IAM user is associated with only one AWS account.

### Groups

A collection of IAM users is an IAM group. You can use IAM groups to specify permissions for multiple users so that any permissions applied to the group are applied to the individual users in that group as well. Managing groups is quite easy. You set permissions for the group, and those permissions are automatically applied to all the users in the group. If you add another user to the group, the new user will automatically inherit all the policies and the permissions already assigned to that group. This lessens the administrative burden.

### Policies

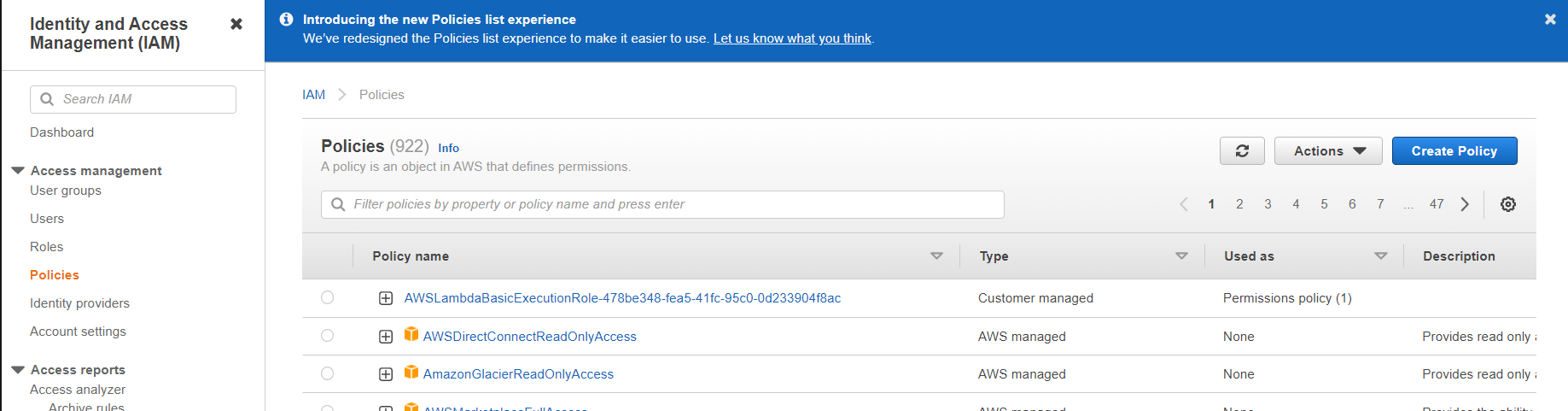
An IAM policy sets permission and controls access to AWS resources. Policies are stored in AWS as JSON documents. Permissions specify who has access to the resources and what actions they can perform.

### Roles

An IAM role is a set of permissions that define what actions are allowed and denied by an entity in the AWS console. It is similar to a user in that it can be accessed by any type of entity (an individual or AWS service). Role permissions are temporary credentials.

**IAM Policies**

1. Login to AWS console management
2. Go to IAM -> Policies -> Create Policy

****

1. Select JSON Tab and paste the below code
2. Click Next: Tags

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents"

],

"Resource": "arn:aws:logs:\*:\*:\*"

},

{

"Effect": "Allow",

"Action": [

"ec2:Start\*",

"ec2:Stop\*"

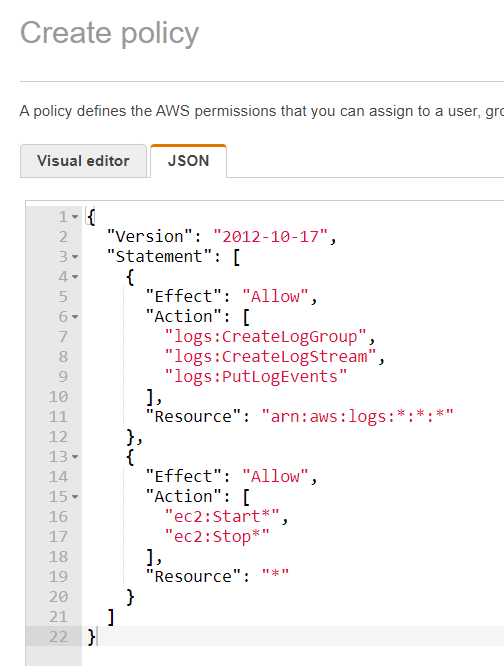
],

"Resource": "\*"

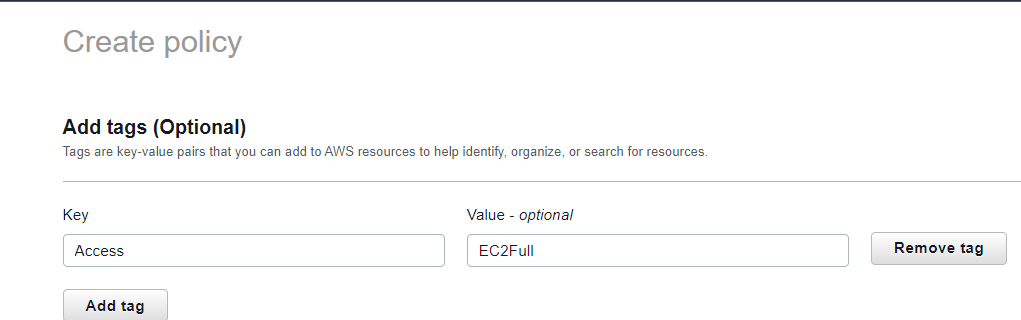
}

]

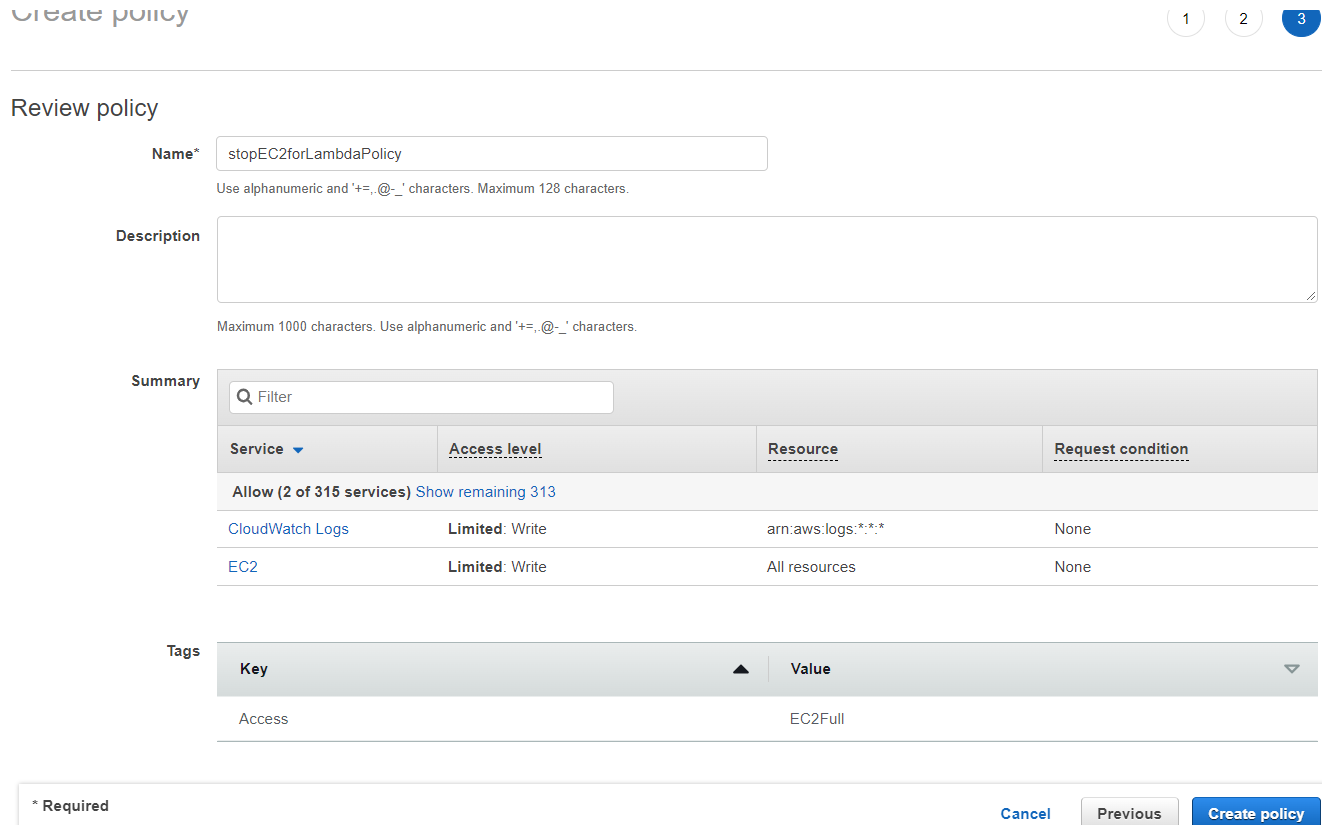
}



1. Next Click Review



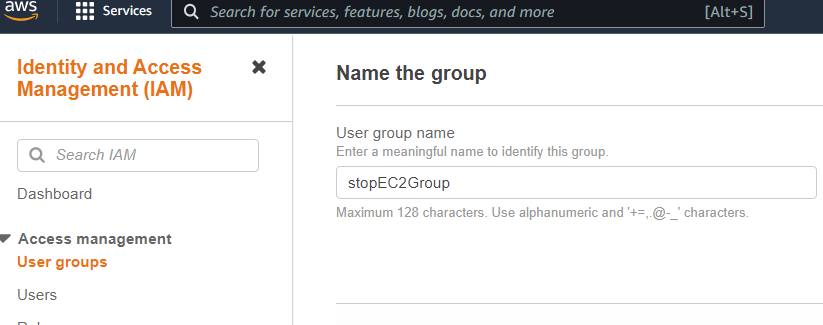
1. Click Create Policy

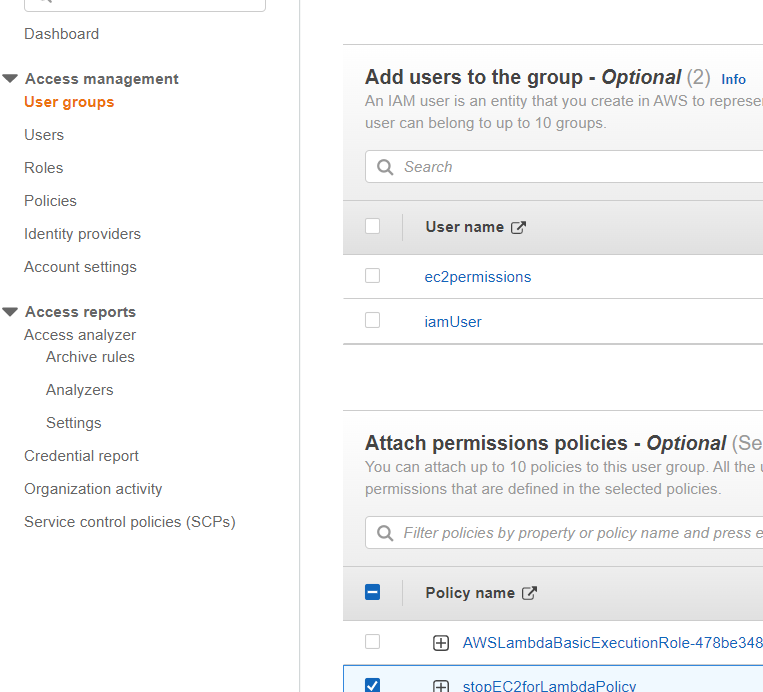


Populated from JSON

**IAM Groups**

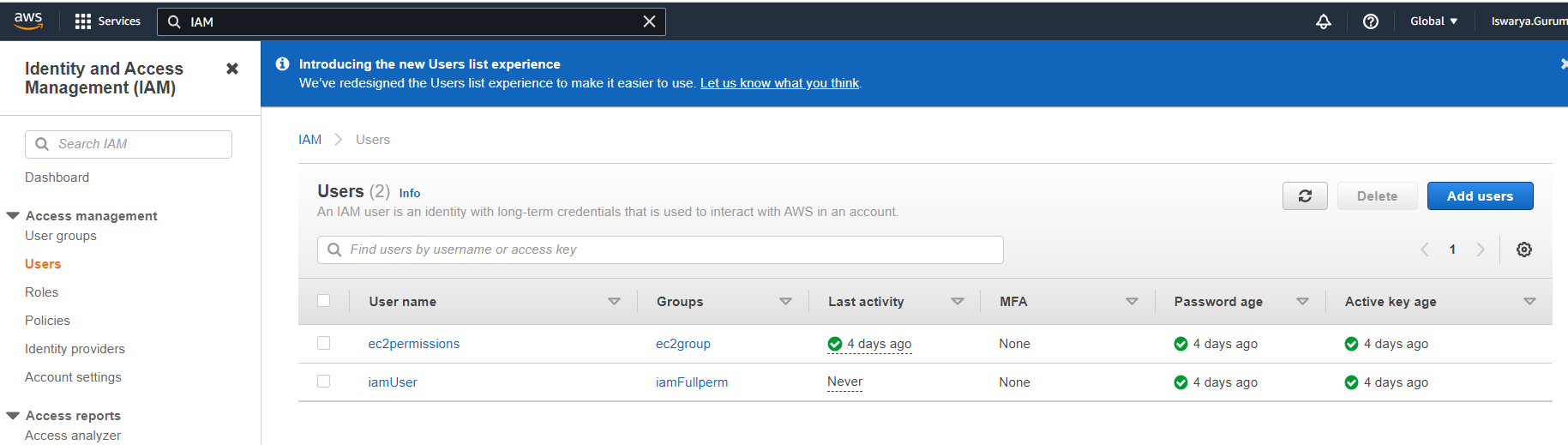
* 1. Go to IAM -> User groups -> Create group



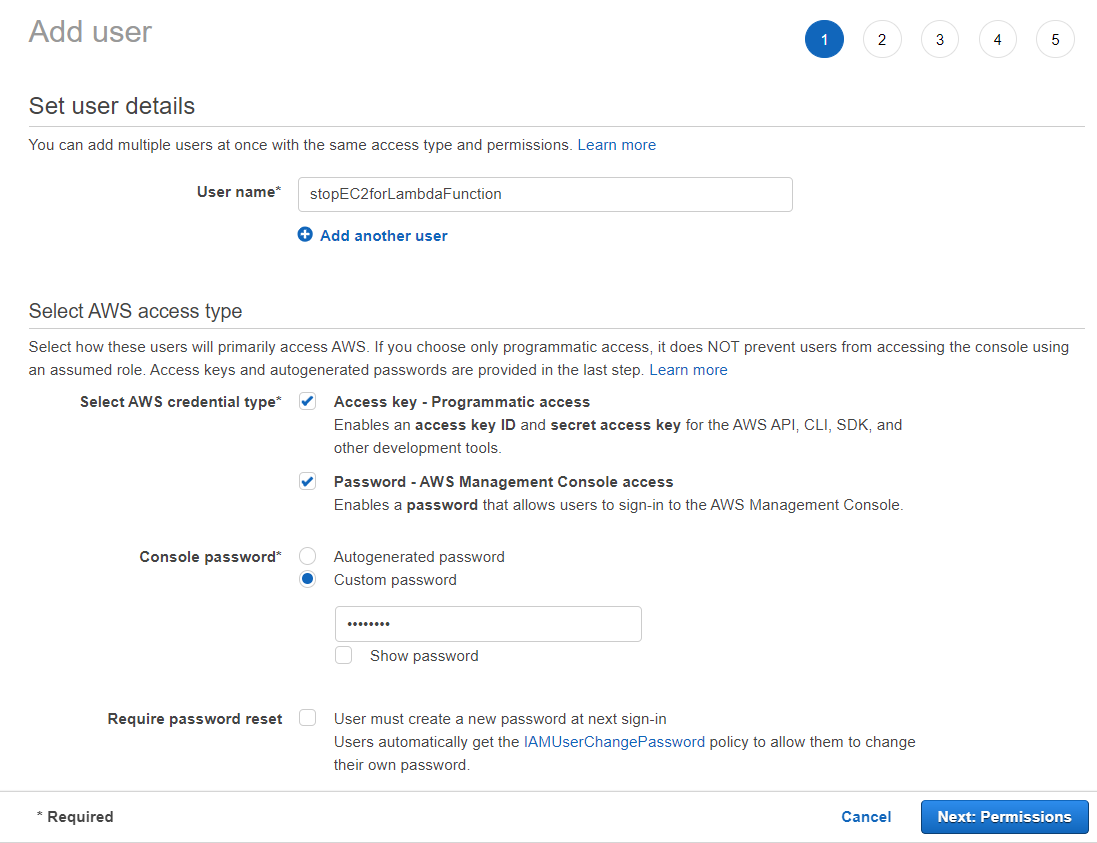
* 1. Select above created policy and click create Group

**IAM Users**

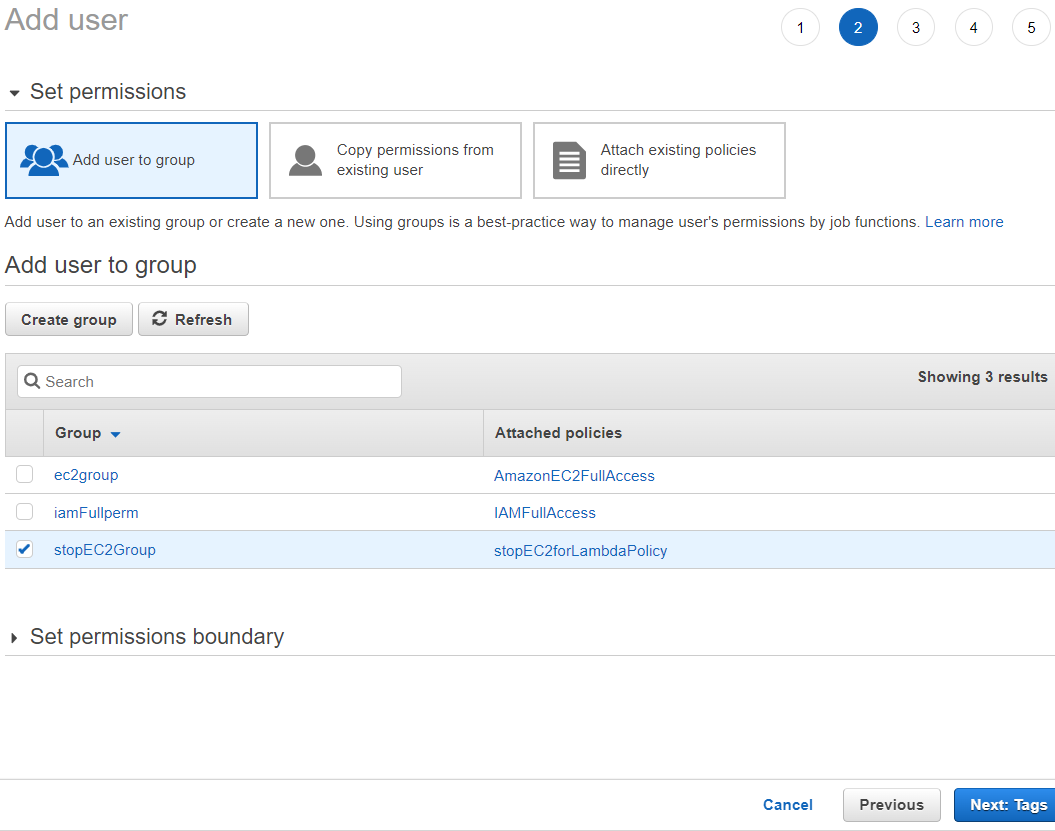
* 1. Go to IAM -> Users -> Add users



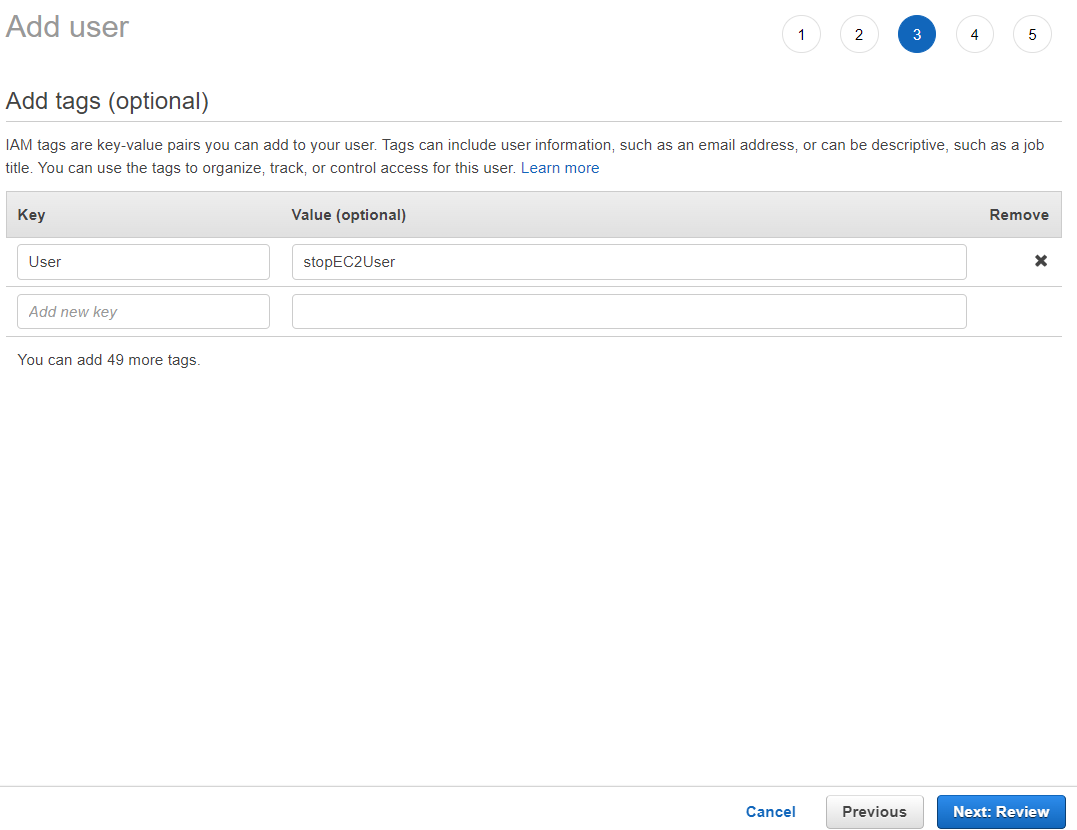
* 1. Set User Name
  2. Select both AWS Access types



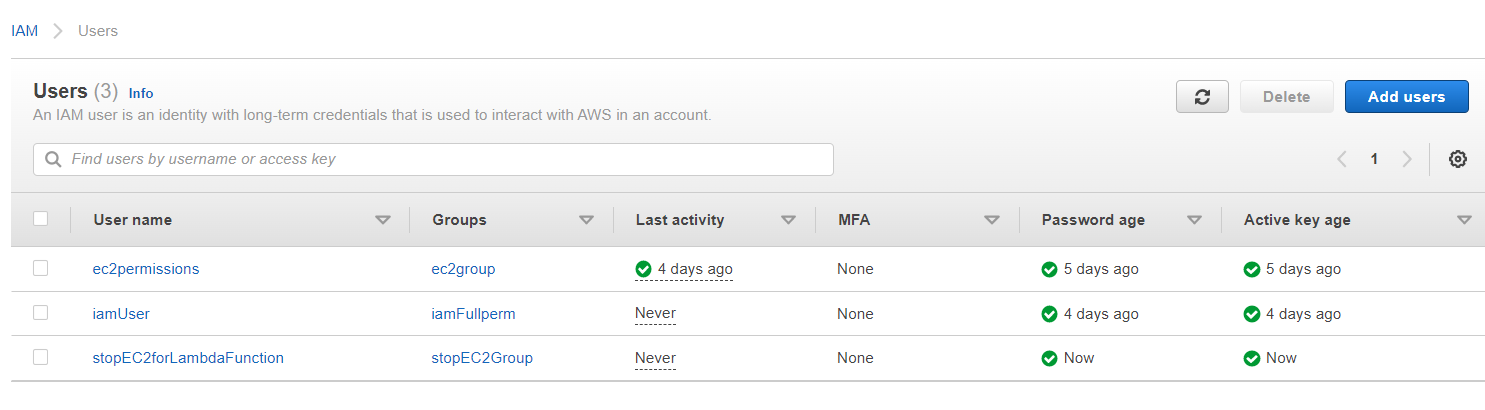
* 1. Add created user to the group



* 1. Add tags and click Next: Review



* 1. Review and Click Create User



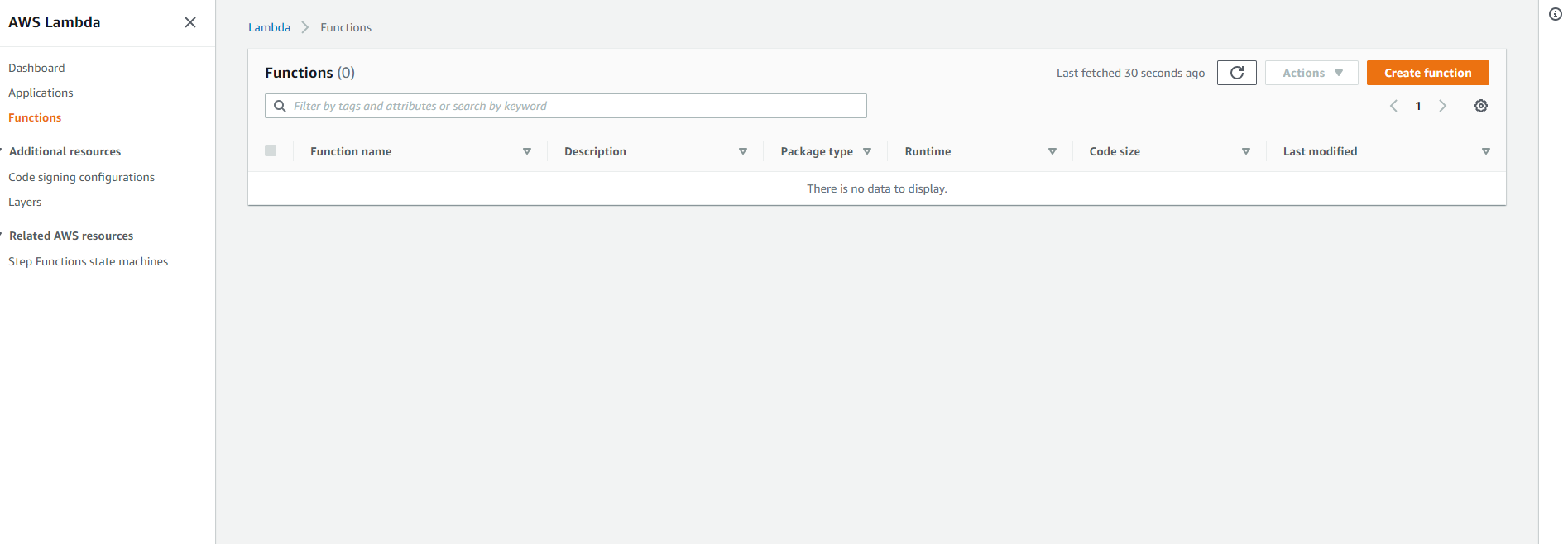
**LAMBDA Function**

* AWS Lambda is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you.

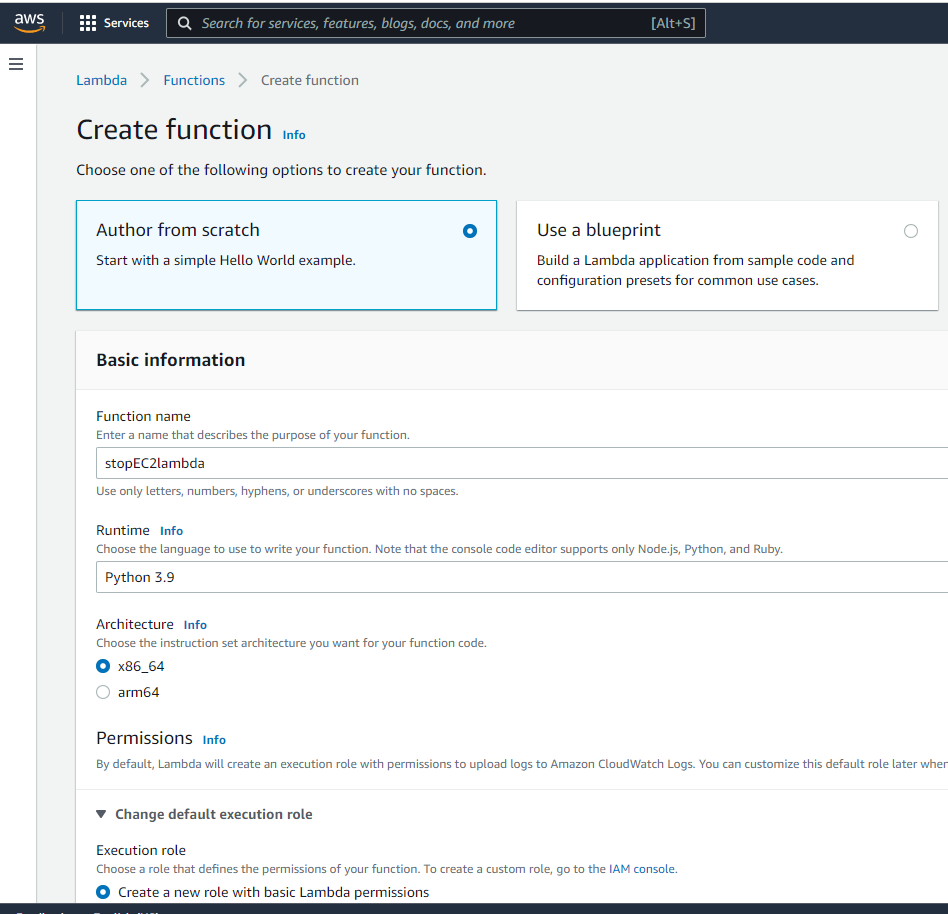
Pricing for Lambda

* There is no additional charge for creating Lambda functions.
* There are charges for running a function and for data transfer between Lambda and other AWS services.

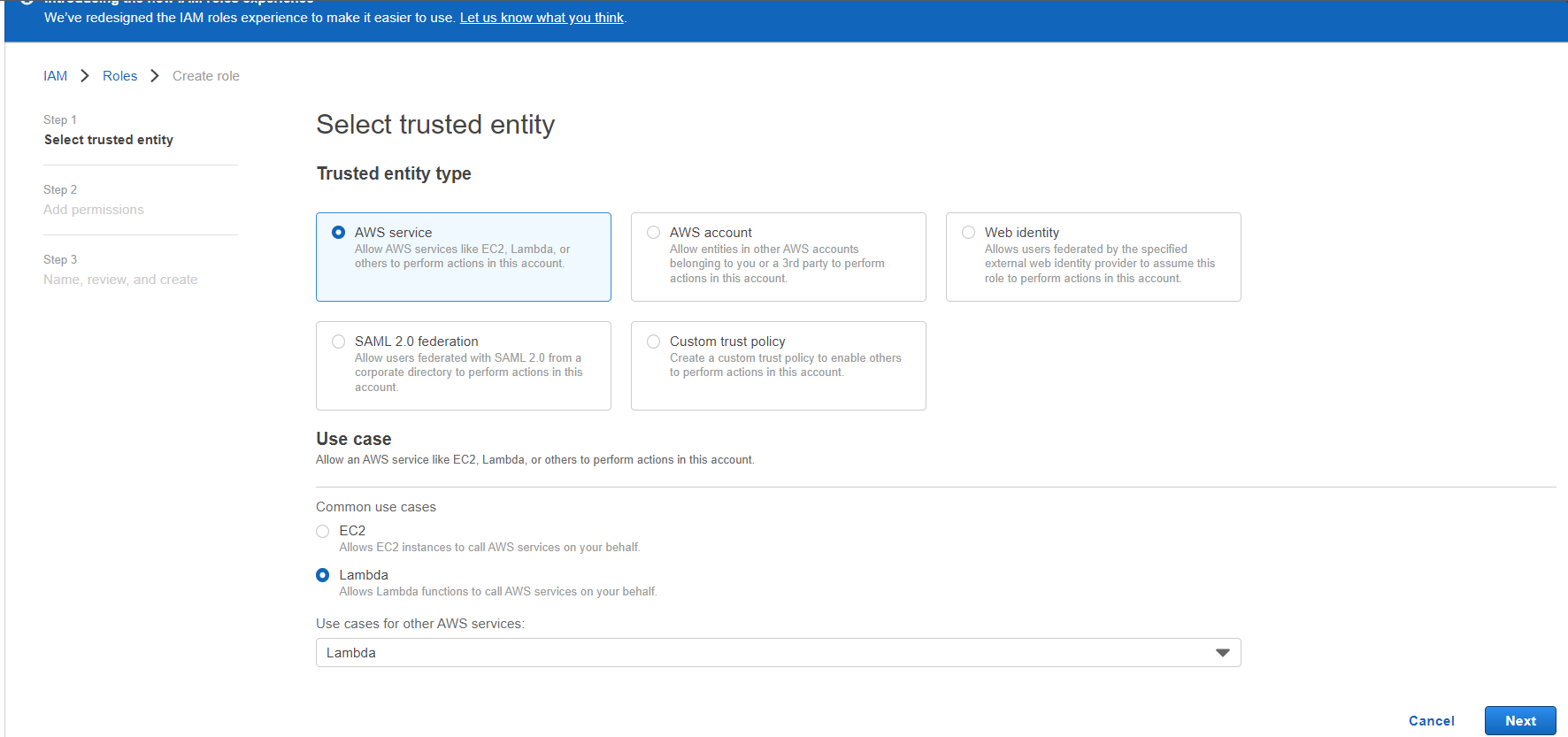
1. Login to AWS management Console
2. Go to AWS Lambda
3. Click Create Function



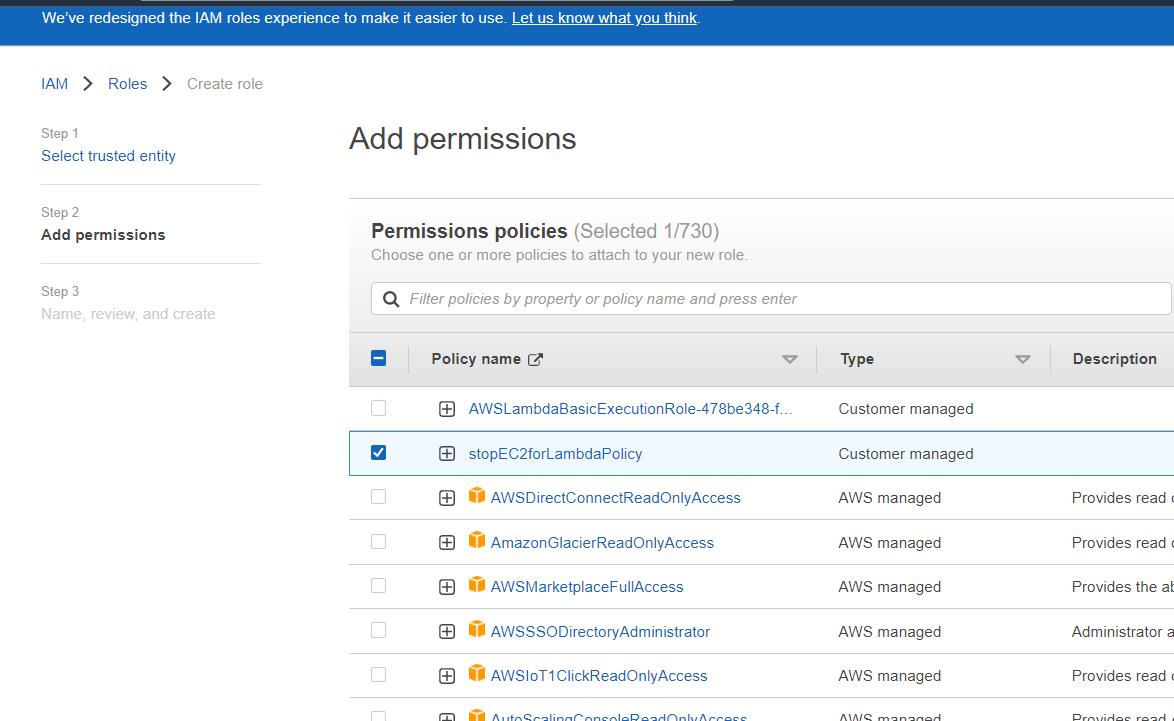
1. Give function name and Runtime programming language.
2. Then click IAM Console



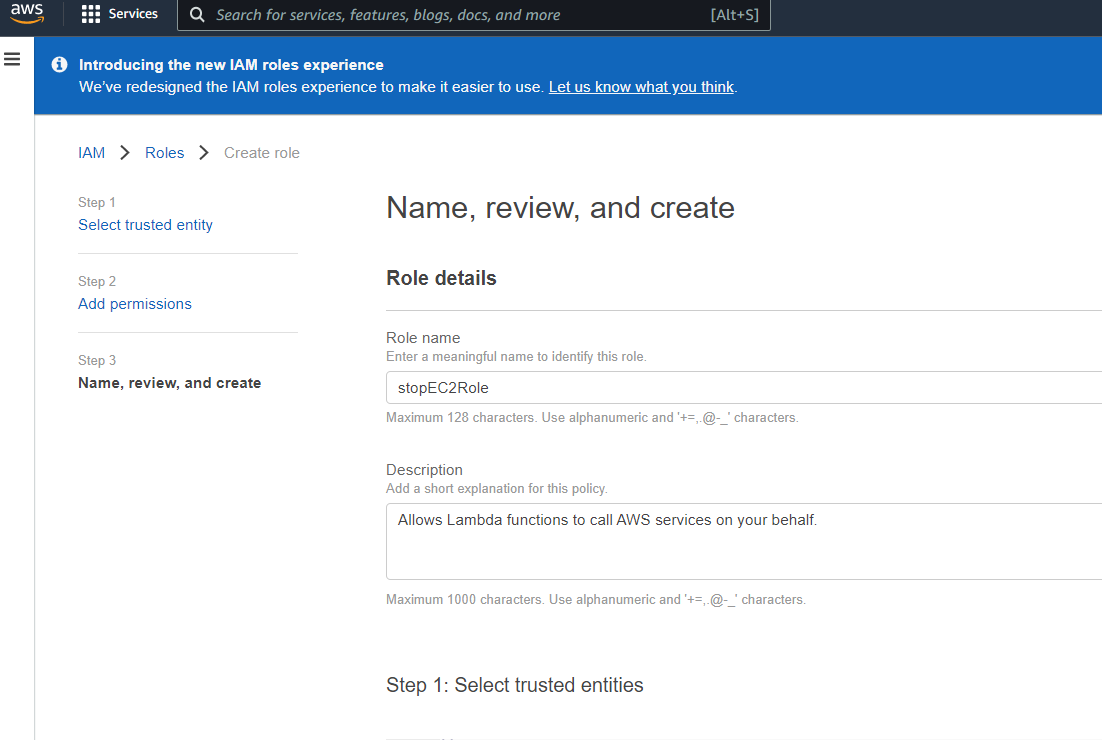
1. Then select AWS service and Use cases as Lambda in IAM Roles
2. Then click Next

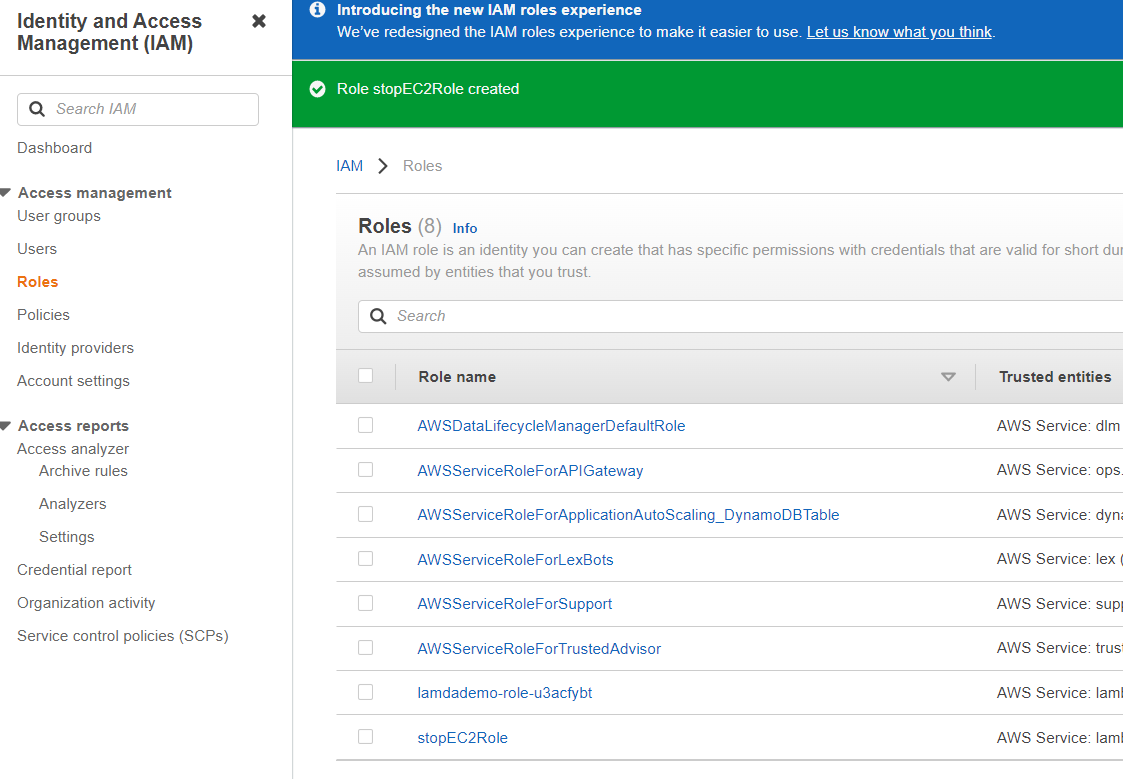


1. Select created policy

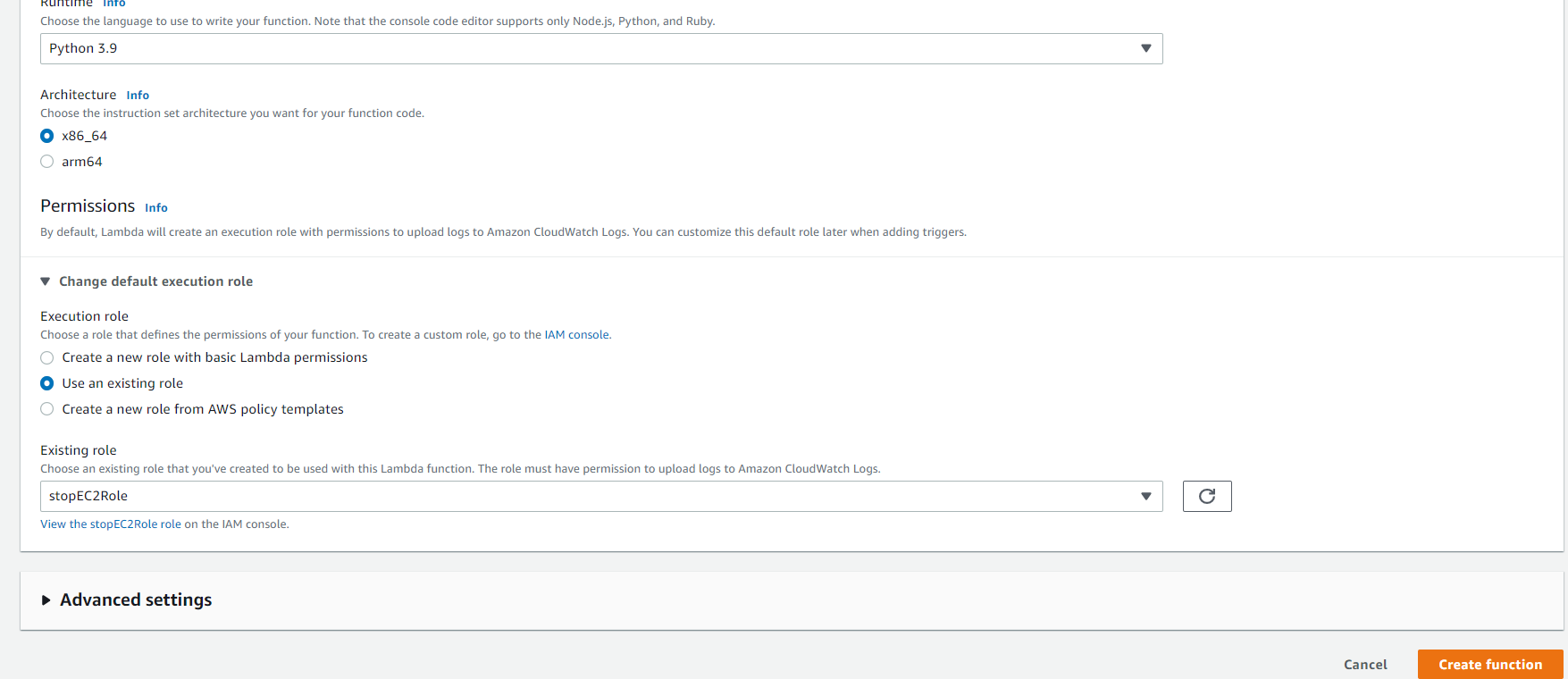


9. Give Role name and click Create Role





* 1. Now choose the existing role in Lambda creation and create function



* 1. Copy the below python coding and replace the region and instance details – Click Deploy

import boto3

region = 'us-east-1a'

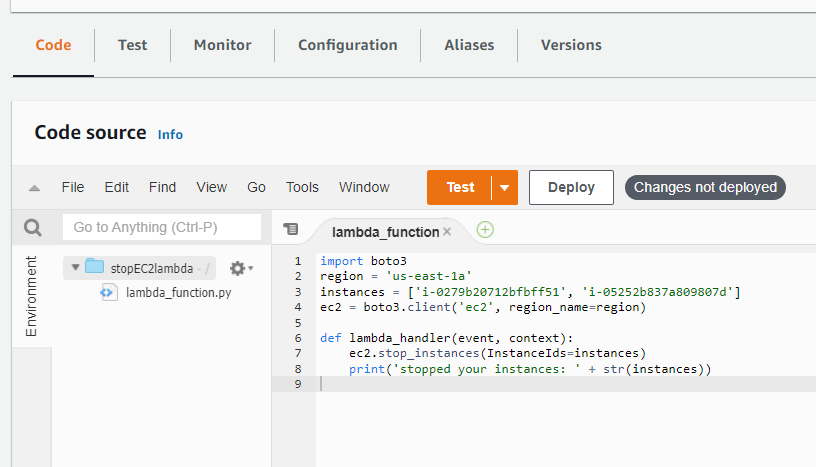
instances = ['i-0279b20712bfbff51', 'i-05252b837a809807d']

ec2 = boto3.client('ec2', region\_name=region)

def lambda\_handler(event, context):

ec2.stop\_instances(InstanceIds=instances)

print('stopped your instances: ' + str(instances))



**Follow step 1 to 10 and start EC2 instance code as follows**

import boto3

region = 'us-east-1'

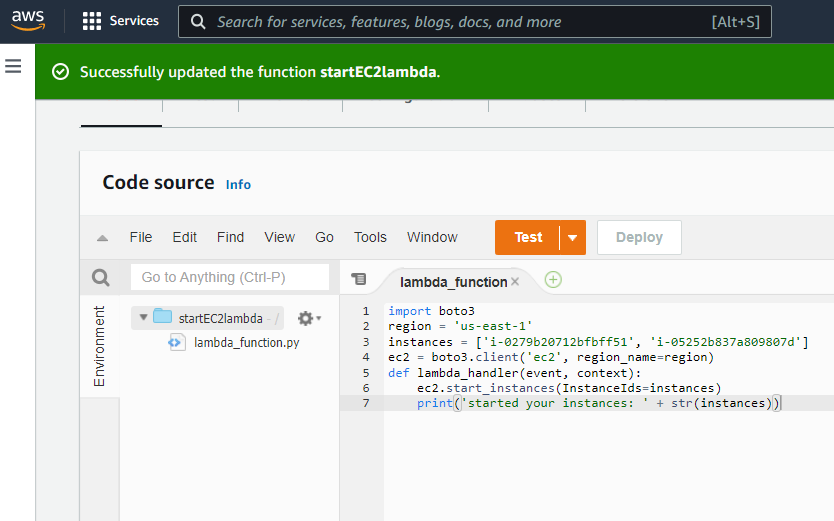
instances = ['i-0279b20712bfbff51', 'i-05252b837a809807d']

ec2 = boto3.client('ec2', region\_name=region)

def lambda\_handler(event, context):

ec2.start\_instances(InstanceIds=instances)

print('started your instances: ' + str(instances))



**Cloud Watch**

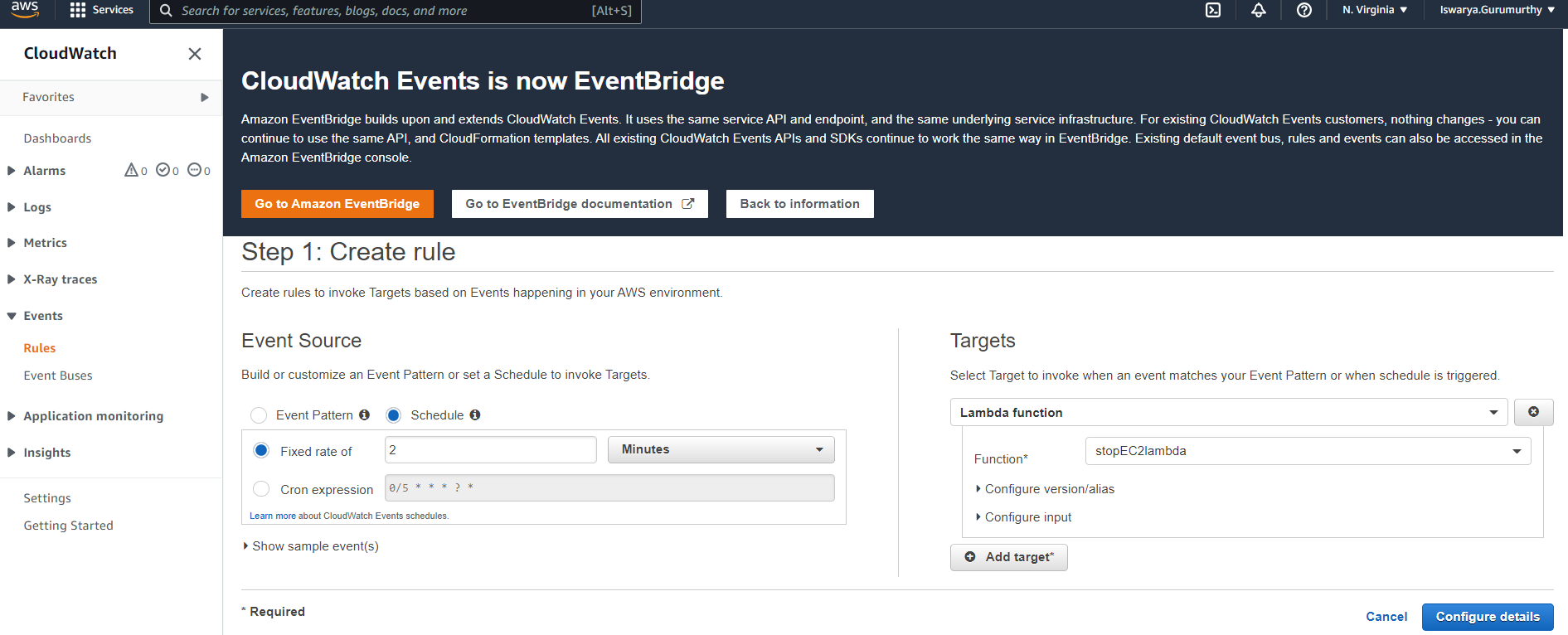
|  |  |
| --- | --- |
| **CloudWatch** | **CloudTrail** |
| * CloudWatch is basically a monitoring service for AWS resources and applications. | * CloudTrail is a web service that is mainly concerned with what is done on AWS by whom. |
| * Tracks metrics and monitor logs. | * Detects user activity and API usage. |
| * CloudWatch delivers metric data in 1 minute period for detailed monitoring and 5-minute periods for basic monitoring. | * CloudTrail delivers an event within 15minutes of an API call. |
| * CloudWatch stores data in its own dashboard in form of metrics and logs. | * CloudTrail centralizes all the logs across the regions and stores them in S3 bucket. |

1.Go to Cloud Watch

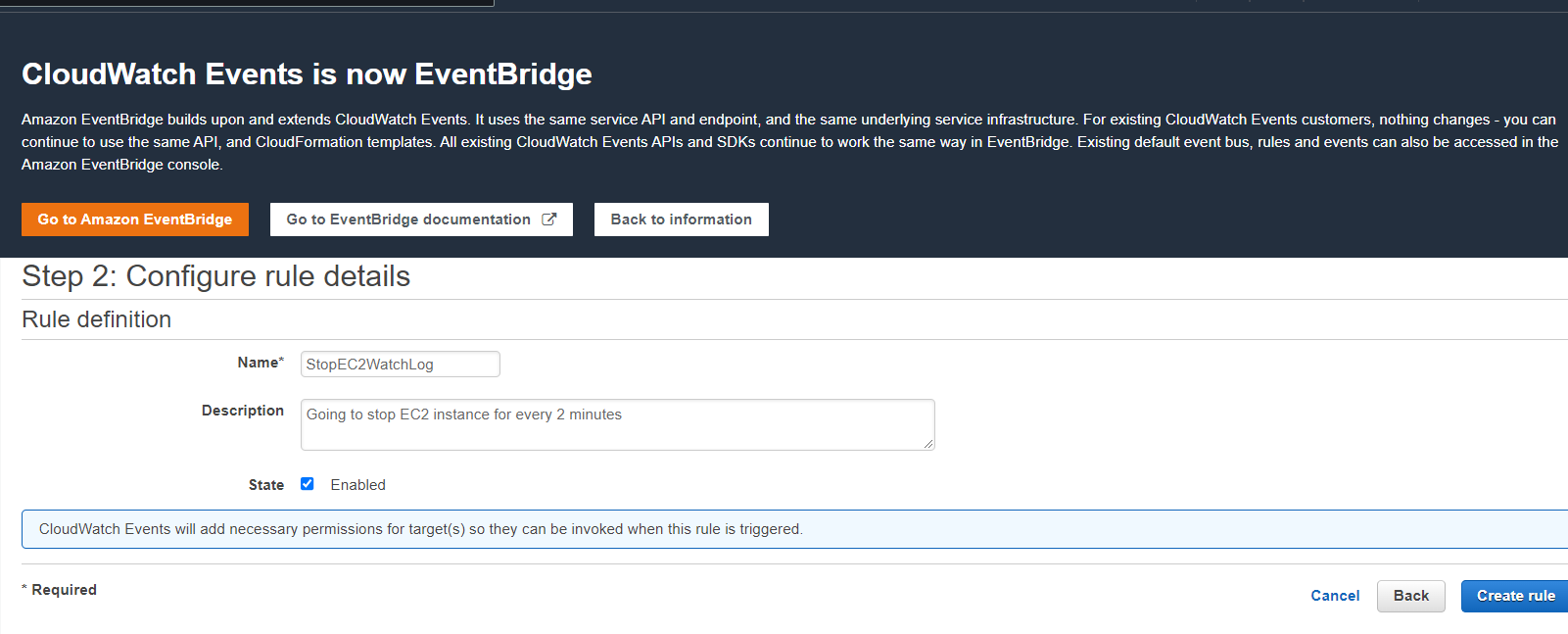


2.Click create rule, select schedule time of 2 minutes and Add Target with the trigger of created lambda function

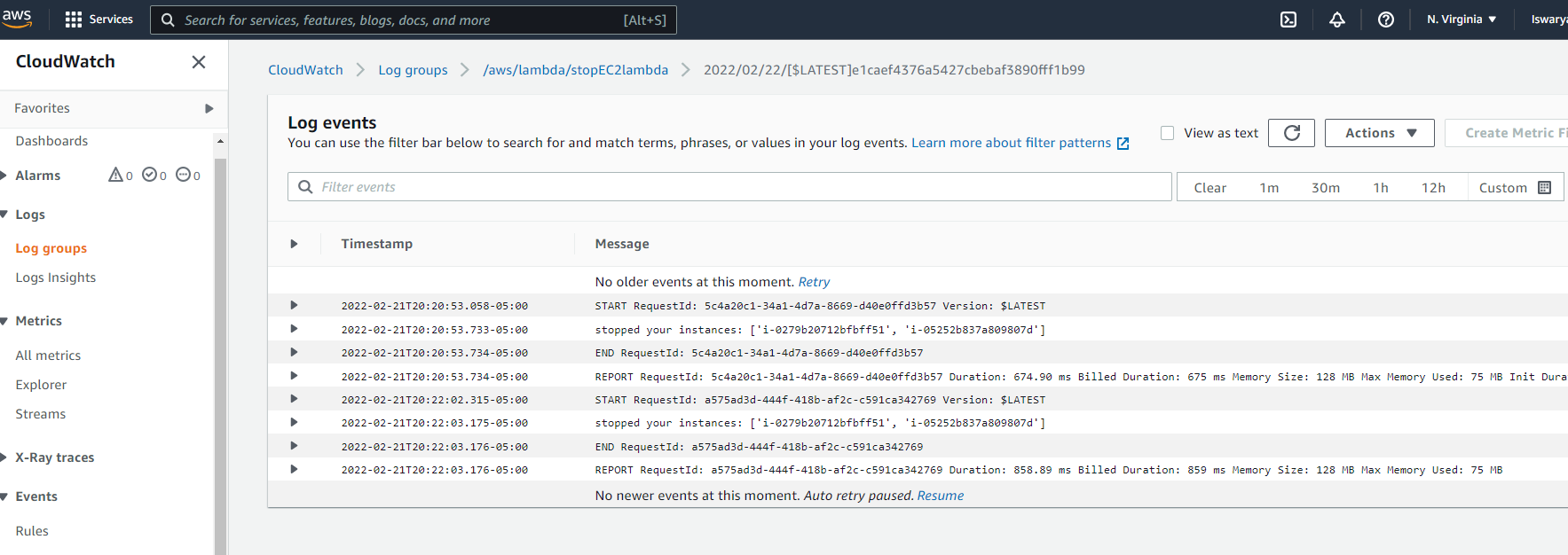
3. Click configure Details

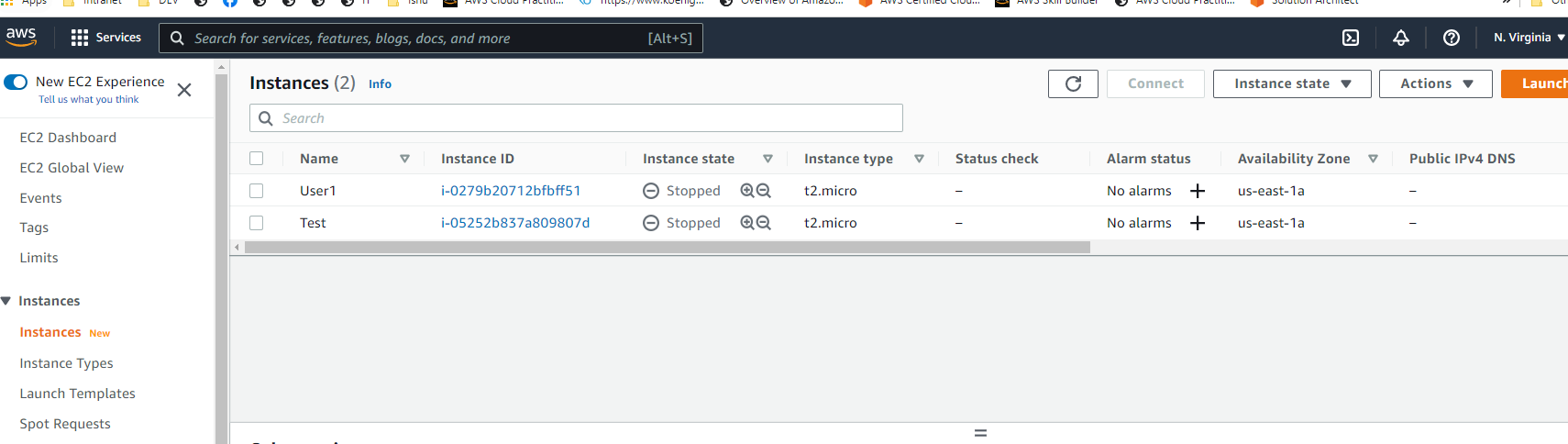


4.Specify name for the rule and click create rule



5.Click log groups and select the rule name to see the logs





**TO Start EC2 instance**

Follow the steps 1,2,3 and 4

